μ mikro számítógép magazin

Ára: 30 Ft

TORONYÓRÁT LÁNCCAL?

1987/8

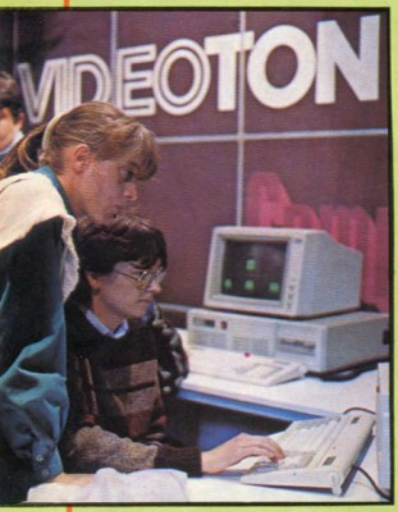
Ez látható többek között az SZKI „standján"

A Novotrade színes pavilonja

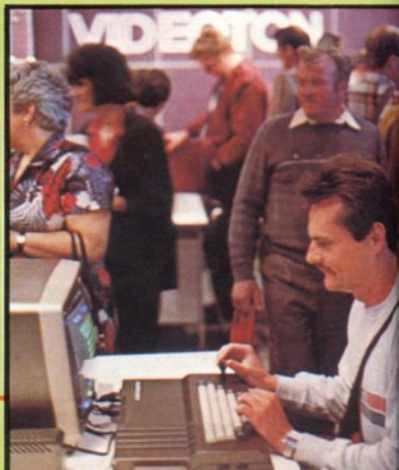
A tavaszi BNV-t,

a műszaki termékek szakvásárát felkeresve a laikusok is tapasztalhatták, hogy változatlanul a magyar ipar egyik legdinamikusabban fejlődő ága az elektronika. A kiállított híradás- és számítástechnikai termékek a szakmai érdeklődőkön kívül óriási közönséget vonzottak, annak ellenére, hogy a majd két hétig nyitva tartó vásárvárosban másféle látnivaló is akadt bőven. Mindenesetre csupán a számítástechnika területéről harminckét magyar és tizenöt külföldi cég által gyártott, illetve forgalmazott különböző kategóriájú számítógépeket, számítógép-hálózatokat és a legkorszerűbb telexberendezéseket állították ki.

(Rövid vásári látogatásunkról a 48. oldalon számolunk be.)



A Videoton sikergyártmányai



5. ÉVFOLYAM
1987/8. SZÁM

μ mikro számítógép magazin

A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG ÉS A KISZ KÖZPONTI BIZOTTSÁG LAPJA

A kiadvány a Tudományszervezési és Informatikai Intézettel együttműködve készül

A szerkesztőbizottság vezetője:
Kovács Győző

E számunkat szerkesztették:

Bakos Tamás
(programozástechnika)

Broczkó Péter
(hírek)

Kovács Győző
(levelezés)

Lindner László
(sakkprogramozás)

Petróczy Judit
(könyvek)

Simonyi Endre
(klub)

Varga András
(iskola-számítógép)

Címképünk:
Ramocsai Imri munkája

μ mikro számítógép magazin

Felelős szerkesztő:
Könyves Tóth Pál

Szerkesztőség:
1027 Budapest, Fő u. 68.
Telefon: 154-250

Levélcím:
1371 Budapest
Pf. 433.

Kiadja az Ifjúsági Lap- és Könyvkiadó Vállalat

Felelős kiadó:
dr. Petrus György
igazgató

Kiadóhivatal:
1065 Budapest, Révay u. 16.
Telefon: 116-660

Terjeszti a Magyar Posta
Előfizethető a hírlapkézbesítő hivataloknál
és a Posta Hírlapelőfizetési és Lapellátási Irodáján
(1900 Budapest V., József nádor tér 1.)
vagy átutalással a 215—96 162 pénzforgalmi jelzőszámra.

Megjelenik havonta
Egy szám ára 30,— Ft
Előfizetési díj:
egy évre 360,— Ft
fél évre 180,— Ft
Külföldön terjeszti a Kultúra,
1389 Budapest, pf. 149.
és a Magyar Média
1932 Budapest, pf. 279.
86—0253

Szikra Lapnyomda
Budapest (87-0995)
Felelős vezető:
Csöndes Zoltán vezérigazgató

INDEX: 25 629
ISSN 0236-6088

Tartalom

Jubileum 2
Fütyül a magyar számítógépes tanulás vonata 8
OKTA-TOTÓ 14
ATARI kontra COMMODORE 64 15
Vigyázat! Tolvaj! 26
Mesterséges értelem IV. 27
Hallásolvasás 31
Az adattípus fejlődése II. 32
μINFORM 35
Adok — veszek — cserélek 37
Elektronikus lemez 42
KANYAR a MARS-on 44
Petike szétszedi apuka óráját 45
Vásári kitekintő 48

ISKOLA — SZÁMÍTÓGÉP

Bemutatjuk az OPI programirodáját 3
Egy egészen egyszerű világmodell 4
Nemes Tihamér 6
A Nemes Tihamér OKSZTV döntője 1987. 7

DIÁKROVAT

Ékezetes karakterkészlet nyomtatóra 10
Kétnapi Tízórai 11
Apróságok 12
Szintetizátor 13

PROGRAMOZÁSTECHNIKA

BASIC és gépi kód 18
Pontosabb számítások Pascalban 19
Z80 programok haladóknak Spectrumra és Primóra 21
Teknősbéka-grafika IV. 23
Toronyórát lánccal . . . 24
. . . Bábel tornyára? 25

μPROGRAMOK

Magyar ékezetes karakterkészlet EPSON FX—100-as és 105-ös nyomtatókra 34

μKLUB

Integrált szoftver 36
Egyszerű, Z80 alapú mikroszámítógép 38
Grafikus nyomtatás K6311 mátrixnyomtatóval 40
Biztos, hogy hibás a számítógépünk? 42
Két érdekes rövid program C64-re 42

SAKKPROGRAMOZÁS

Bitek és figurák 43

AZ OLVASÓ ÍRJA 46

KÖNYVEK 46

HÍREK, ÉRDEKESSÉGEK 47

„A hatalmas elme s mélységes tudás.
E kettő emel fel, nincs még semmi más."
(Jusuf Hasz Hadzsib: A boldogító tudomány, VI. fejezet. Brodszky Erzsébet fordítása)

Jubileum

Április közepén rövid telefonértesítést, majd 30-án expresszlevelet kaptam:

„A tervmatematika szak megalakulásának 25. évfordulója alkalmából merült fel az a gondolat, hogy rendezzünk találkozót a szak végzett hallgatói részére. A gondolat most valóra válik, május 9-én szombaton 17 órakor az egyetem aulájában rendezzük a találkozót. Összejövetelünk teljesen informális, néhány köszöntő szón s a szakosok körében végzett kérdőíves felmérés kiértékelésének rövid ismertetőjén túl csakis beszélgetés, no meg hidegtálas vacsora lesz.

A találkozóra ezúton szeretettel és tisztelettel meghívjuk.

A Szervezők"

Az ember mindig elfogódott, ha a múltjának egy darabjával szembesül. Az első gondolatom, amikor az értesítést megkaptam, az volt: el sem hiszem, hogy már 25 (pontosabban 27) év telt el azóta, hogy Európában talán a legelsők között a közgazdasági egyetemen olyan új típusú szakemberek képzése kezdődött meg, akik értették, sőt megértették a gazdasági problémákat, emellett alapos matematikai ismerettel és ráadásul az akkori szintnek megfelelő számítástechnikai képzettséggel is rendelkeztek.

Az ügy első számú harcosa Krekó Béla volt, aki igen sok támogatást kapott László Imrétől, az egyetem oktatási ügyekért felelős rektorhelyettesétől. A matematika tanszék tanárai — Halmai Erzsébet, Bikics Istvánné, Meszéna György — gondoskodtak arról, hogy a hallgatók a gazdasági élet adta feladatokat ne csak kvalitatív, hanem kvantitatív módon is meg tudják oldani.

Én, mint olyan sokszor az életben, ismét szerencsés voltam. Akkor éppen túl voltunk az M—3 építésén (akik nem tudnák, ez volt az első elektroncsöves számítógép, 1959-ben készült el), jómagam pedig éppen befejeztem egy kéziratot, amely egyrészt alapfokú elektrotechnikai ismereteket adott, másrészt az új számítógép működését mutatta be. Amikor a tervmatematika szak alakulgatott, én — kezemben a kézirattal — éppen azt az egyetemet kerestem, ahol megengednék az akkor kibernetikának nevezett „tudomány" tanítását. Így Krekó Bélának nem kellett kétszer mondania; hívását elfogadtam, és 1960 őszén elkezdhettem a leendő tervmatematikusoknak a hardvertanítást. 6 év múlva hagytam abba, amikor az informatika tanszék megalakult. Nagy bánatom, hogy ez az első két számítógépes jegyzetem elveszett. De talán valamelyik diákom megőrizte; reménykedem, hogy egyszer sikerül valakitől megszereznem.

A találkozón a volt hallgatókkal együtt idéztük fel az első éveket. 14-en voltak az úttörők, az első évfolyam válogatottan jó diákjai, de utódaik sem voltak gyengébbek — legalábbis, amíg én ott tanítottam, így volt. Hangulatos órákra, még hangulatosabb tanulmányi kirándulásokra emlékeztünk, Pécsre, Szegedre, Egerre; az éjszakába nyúló beszélgetésekre, amelyek emléke jólesően sokakban él.

A találkozó szervezőinek szándéka utat talált: valóban sikerült informális összejövetelt rendezniük. Az alaphangot már a „néhány köszöntő szó" is megadta, amelyet Zalai Ernő, a tervmatematika szak mai professzora — ő maga is 1962-től volt itt diák — mondott, keresetlen egyszerűséggel idézve diákéletünk hangulatát.

Meszéna György is a tőle megszokott humorral emlékezett az elmúlt évekre. Körülbelül 600 diák végzett a tervmatematika szakon — mondotta —, amelynek a neve kétszer is megváltozott. Az indulás után mintegy 6 évvel Népgazdaság Tervező Szakon végeztek a hallgatók, majd újabb 8 év múlva a Tervgazdasági Szak eredményes befejezéséről vehették át a diplomát. Most ismét változtatásokat terveznek, amivel kapcsolatban nagyon halkan többen is megjegyeztük, hogy talán ideje volna visszatérni az eredeti névhez, a technika és a tudomány mai állásának megfelelő tartalommal.

A találkozón résztvevő kb. 300—350 volt hallgató megerősítette Meszéna György véleményét: a szakot helyes volt létrehozni, az itt végzettek megállták a helyüket. Nem egy közülük ma már a szakma megbecsült vezetője. Bebizonyosodott az is, hogy egy közgazdász számára nem hátrány, ha a problémákat kvantitatív módon, informatikai eszközökkel is képes megoldani.

A találkozó szervezői a végzett hallgatóknak kérdőíveket küldtek szét. Dr. Vastag Gyula, aki 1979-ben és Kovács Erzsébet, aki 1978-ban végzett, értékelte ki a 158 visszaérkezett választ. Szíves hozzájárulásukkal néhány érdekes véleményt és következtetést közzéteszek.

Körülbelül egyforma számban küldtek választ a három korszak (tervmat, népgazd. terv és tervgazd. szakok) hallgatói. A válaszokból kiderült, hogy a hallgatók főleg gimnáziumból jöttek (83%) és elsősorban azért, mert szerették a matematikát (82%). A másik jelentős motiváció: véleményük szerint egy „elit" szakra jelentkeztek. Nem lehet véletlen, sem mellékes, hogy sokan (78%) ezt válaszolták: ha újra kezdenék a pályát, ugyanezt a szakot választanák! Ez a vélemény biztos dicsérete az iskolának.

Gazdasági helyzetünk bírálatának is tekinthető, hogy az e szakon végzett hallgatók szinte mindegyike kutatóintézetben, főiskolán, egyetemen, és nem a vállalati szférában dolgozik. Vajon a vállalatoknak ma még nincs szükségük matematikában és számítástechnikában jártas közgazdászokra?

Arra a kérdésre, hogy mennyire kötődnek a munkahelyükhöz és mi motiválja őket az ottmaradásban, a legtöbben azt felelték, hogy általában elégedettek, kielégíthetik tudományos ambícióikat, a munkájuk végzéséhez elegendő önállóságot kapnak. Csak néhányan válaszolták azt, hogy máshol sem jobb, vagy hogy lakást kaptak, esetleg hogy a magasabb fizetés a megtartó erő.

Megkérdezték a volt hallgatókat a szak jövőjéről is. Érdekes vélemények hangzottak el, amelyekre az egyetemi döntéshozók is bizonyára odafigyelnek. Abban mindenki egyetértett, hogy változatlanul sok és kemény matematikát kell oktatni; egyetlen hallgató sem bánta meg, hogy alaposan felkészítették ebből a tárgyból. Egyesek azt javasolták, hogy a szak legyen gyakorlatibb: adjon több vezetői ismeretet, mai modern számítástechnikai tudást. Oktasson például adatbázis-kezelést, ismertessen olyan alkalmazói rendszereket, amelyeket ma az egész világon széleskörűen alkalmaznak.

Számomra a késő éjszakáig tartó beszélgetés leginkább azt bizonyította, amit 1960-ban talán nem is sejtettünk, hogy az első 14 hallgató elindítása új korszakot nyitott a hazai közgazdászképzés történetében. Ennek jelentősége csak ma, az informatika társadalmi méretű elterjedésének korában mérhető fel igazán.

További sok sikert kívánok, kedves tervmat'-os barátaim! KOVÁCS GYŐZŐ

Bemutatjuk az OPI programirodáját

1986 májusában az OPI-n belül önálló kutatás-fejlesztéssel foglalkozó részleg alakult: a Számítástechnikai Programiroda. Szücs Barnát, az iroda vezetőjét kérdeztük tevékenységükről.

— Irodánk informatikai szemléletű számítástechnikával foglalkozik. Segíti a kormányprogram közoktatásra háruló feladatainak megvalósítását. A végső cél, hogy az iskolákban meghonosodjon a számítógépek és a hozzájuk tartozó módszerek alkalmazása. Az 1990-ig terjedő időszakra hat feladatrendszert, témát dolgoztunk ki.

Az első téma: *A számítástechnikai alapműveltség tartalmának, összetevőinek és követelményeinek meghatározása*

A kérdésről szakértői vélemények születnek, amelyeket szélesebb körben megvitatnak. A végeredmény várhatóan 1988-ban egy nyomtatásban is megjelenő tanulmány lesz.

— Ez a kötet tartalmazza az iroda végső következtetését?

— Nem. Az OPI nem kíván levonni végső következtetést, a tanulmánykötetet a szemléletet befolyásoló kiadványnak szánja. Ez jelenleg nemcsak Magyarországon, hanem szinte mindenütt a világon a legvitatottabb témák közé tartozik. Az UNESCO is konferenciák sorát szervezte már ebben a kérdésben.

Nálunk — néhány technikusképző szakközépiskola kivételével — a számítástechnika nem tantárgy. Ennek okai közül csak néhányat említek. Nehéz lenne az amúgy is túlzsúfolt tantervekbe a sok tantárgy közé újat beilleszteni. Nincsenek meg továbbá a tárgyi és személyi feltételek: gépek és tanár nélkül nem lehet számítástechnikát oktatni! Nem kedveznek az ügynek az olyan elvi viták sem, mint például hogy melyik fontosabb: a számítástechnikai alapismeretek vagy az alkalmazói kultúra terjesztése.

— Egyesek azt állítják, hogy a kialakult ösztönös folyamatot megöljük a szabályozással.

— Ezzel nem értek egyet. Önmagában az, hogy a folyamat spontán jön-e létre vagy központilag irányítják, nem ad garanciát a minőségre. A kettőnek kölcsönösen hatnia kell egymásra. Az ösztönös kezdeményezésekkel számolni kell: az okos központi tervezés mintegy „meglovagolja" azokat. Napjainkban a számítástechnikai alapképzés — a francia, szovjet és bolgár gyakorlat kivételével — mindenhol decentralizált, tehát ez nem magyar probléma.

A második téma: *Tanulóközpontú és rendszerbe foglalt oktatási programcsomagok kidolgozása és kipróbálása*

Ennél a témánál különösen a fakultációs képzésre és a szakköri tevékenységre alapozunk. Szerkezetileg más-más megoldást választunk az általános és a középiskolák számára.

Az iroda pedagógiai programokat, segédleteket és hozzájuk tartozó szoftvereket dolgoztat ki, amelyek kereskedelmi forgalomba kerülnek. Ezek modulrendszerű csomagok, és célkitűzés, tananyag, gépi eszköz szerint variálhatók. Elsősorban az általános iskolákban szeretnénk ezeket elterjeszteni. A megoldás egyébként hiánypótló és egyedi: külföldön sincs ilyen.

A középfokú oktatásban három kérdésre keressük a választ: milyen lenne a gimnáziumban a számítástechnikai tagozat és milyen egy szakmai előkészítő pedagógiai program, továbbá hogyan festene az „életre felkészítő" (adat-előkészítő, szövegszerkesztő) képzés? Bármelyik megvalósítható fakultáció keretében. Ilyen fakultáció már működik a pécsi Apáczai Csere János Gimnáziumban. Az oktatási programcsomagok kidolgozásánál kérdés, hogy érdemes-e azokat a jelenlegi gépparkra készíteni? Számolni kell ugyanis azzal, hogy a PC-k az iskolákban is egyre nagyobb számban terjednek.

A harmadik téma: *A számítástechnika beillesztése a tantárgyi rendszerbe*

Az iroda ezen a területen több lépésben tervezi a fejlesztést. Az első lépés tantárgyak szerint számba venni, hogy milyen oktatóprogramok állnak hozzájuk rendelkezésre és milyen újakat érdemes kidolgoztatni.

A második lépésben a kijelölt fejlesztési pontokat rendszerezzük, végül pedig pedagógiai forgatókönyvet készítünk a szoftverek kidolgozására. Ezt követi az iskolai ellenőrzés és hasznosítás. A fejlesztési folyamat része az iskolai kipróbálás.

— Milyennek ítéli az oktatóprogramok jelenlegi helyzetét?

— Pillanatnyilag nagy az összevisszaság, ezért is jelöltük ki ezt a témát kutatási-fejlesztési iránynak. Egyes területekre, például a függvénytranszformációra, feleslegesen sok az oktatóprogram. A valószínűségszámításra, a halmazelméletre kevesebb készült. Mennyiségileg és minőségileg is vizsgálni kell a szoftvereket. Egyik-másik például pedagógiai szempontból káros.

— Hogyan lehet káros az oktatóprogram?

— Például az ellenőrzés, vizsgáztatás túlburjánzásához vezet, egyszerűen csak drillezteti a tanulókat. Káros az oktatóprogram akkor is, ha helyettesíteni próbálja, feleslegessé teszi a természettudományi kísérleteket. Ilyenekre a geometriai, fénytani programok között találhatunk példát.

Téves nézet, hogy az algoritmizálás önmagában pedagógiai értéket jelent. Tulajdonképpen a tantárgyi témák jó része algoritmizálható. Azt kell vizsgálni, hogy mi az a sajátosság, újszerűség, hatékonyságnövelő módszer a programban, ami segíti a tanulást. Ilyen előrelépést jelent a méréstechnikában az adatfelvétel és a feldolgozás, vagy a folyamatirányítás és -vezérlés. Ez a megismerés új kultúrája.

A negyedik téma: *Szociológiai felmérés*

Óralátogatások, interjúk képezik ehhez az alapot. Azt vizsgáljuk, hogyan hat a számítástechnika a tanítás-tanulás hatékonyságára. Befolyásolja-e a tanulók életmódját és személyiségük fejlődését? A vizsgálat eredménye csak kortörténeti dokumentumnak tekinthető, mert a fejlődéssel együtt a vizsgált közeg is változik.

Az ötödik téma: *A számítástechnikai eszközök iskolai alkalmazásának ergonómiai vizsgálata*

Ez a figyelemfelhívó kutatás a számítástechnikai eszközöknek a tanulók és tanárok egészségére gyakorolt hatását tanulmányozza. Kitér a számítógép-konfiguráció elhelyezésére, a bútorok kialakítására, a helyes megvilágításra és még sok más olyan dologra, amelynek nem tulajdonítunk kellő fontosságot. A diákok helytelen szemlélete az őket érintő egészségügyi kérdésekben élő probléma, komolyan kell foglalkoznunk vele.

A hatodik téma: *Kisiskolás korban a számítástechnika kapcsolata a nyelvi kommunikációs kísérlettel*

Ebben a kísérletben a számítógépet, mint a kommunikáció eszközét nézzük. A Zsolnay-féle elképzelés szerint folyó vizsgálatban a gép által biztosított lehetőségeket tárjuk fel a diák—diák és tanár—diák kapcsolatában.

Az iroda tevékenysége abban is újszerű, hogy kutatási eredmények hasznosítását menedzseli. Lényegesnek tartom, hogy a kutatások eredménye és hasznosítása szorosan összefonódjék.

PINKE GYÖRGY

Egy egészen egyszerű világmodell

Az általános és középiskolában egyaránt szóba kerülnek a populációk életének törvényszerűségei. Hajlamosak vagyunk azonban az embert csaknem teljesen kiemelni ebből a biológiai rendszerből, közben pedig alig vesszük tudomásul, hogy alkalmasan választott modellek segítségével az emberiséget ugyanolyan jól (vagy legalábbis alkotó vitákra indítóan) vizsgálhatjuk, mint az egyszerűbbnek látszó állati populációt. Ráadásul az emberiség vizsgálatakor olyan paraméterekkel és korlátozó tényezőkkel dolgozunk, amelyek mindenkinek kézzelfoghatóak, és melyekről valamennyiünknek van véleménye, tapasztalata.

Természetesen a komoly világmodelleknek és az ezzel kapcsolatos kutatásoknak van múltjuk. Itt most nem célunk az, hogy ezt a tudományterületet gazdagítsuk, hanem, hogy a már ismert modelleket felhasználva megpróbáljunk összeállítani egy olyan egyszerű világmodellt, amely az iskolában jól használható, új gondolatokat ébreszt, és nehezen érthető törvényszerűségeket hűen bemutat.

A modellezés alapja

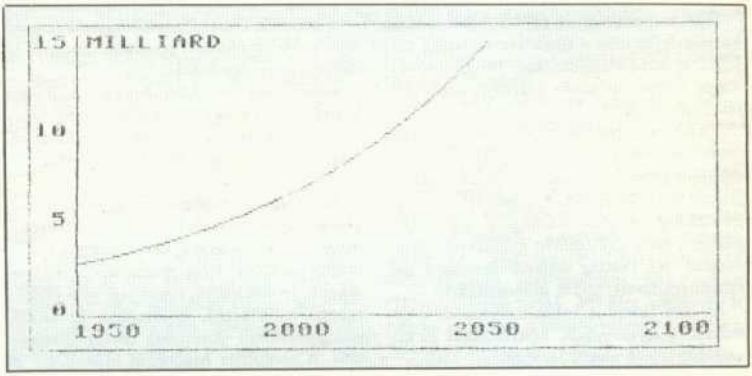
Mint minden ilyen jellegű modellnél, itt is az eddig megismert törvényszerűségekből indulunk ki. Az ismert demográfiai adatok alapján az 1900 és 1980 közötti időt figyelve a Földön a népesség növekedése (a Földön ma mintegy 5 milliárd ember él, évente csaknem 100 millióan születnek. A szerk.) jó közelítéssel egy exponenciális függvénnyel írható le, melynek alakja a következő:

$$dN/dT = (Ksz - Kh)\,N \qquad (1)$$

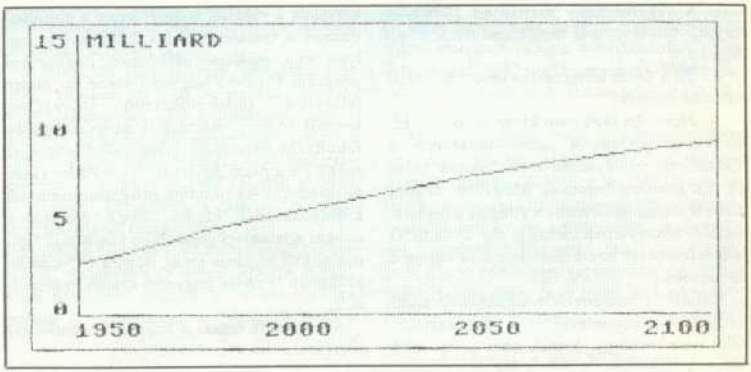
ahol N jelenti az emberiség lélekszámát, Ksz a születési rátát, Kh a halálozási rátát és T az időt.

A növekedés korlátai

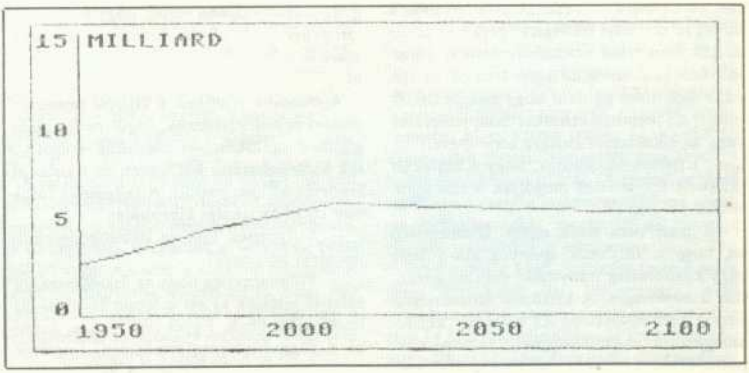
Ez a növekedési ütem azonban már nem tartható fenn hosszabb ideig, mert csökken majd. Ennek tudatában próbáljuk meg összeszedni, hogy melyek azok a korlátozó tényezők, amelyek a modellbe könnyen beépíthetők és valóságos tartalmat hordoznak. Természetesen nemcsak ezek lehetnek a paraméterek és a figyelembe vett hatások. Itt mindössze azt szeretnénk, ha példát adhatnánk a modellezési folyamatra, ami alapján mindenki felépítheti saját modelljét. Az általunk bemutatott feltevések nem szentenciák, lehet, sőt kell is vitatkozni, hiszen ezen a területen kevés a teljes bizonyosság. És ez a vita egyre jobb modelleket szülhet.



1. ábra



2. ábra



3. ábra

A gátló tényezőket olyan formában vegyük figyelembe, hogy a már említett exponenciális növekedési egyenlet (1) születési

és halálozási rátáját módosították a bekövetkező hatások, vagyis:

$$Ksz = Ksz(0) - Ksz(1) - Ksz(2) - \ldots - Ksz(i) \quad (2)$$

és

$$Kh = Kh(0) + Kh(1) + Kh(2) + \ldots + Kh(i) \quad (3)$$

ahol Ksz(0) és Kh(0) jelentse a tapasztalt exponenciális növekedéshez tartozó születési és halálozási rátákat, míg Ksz(i) és Kh(i) azt a születésiráta-csökkenést, illetve halálozásiráta-növekedést, amit az i-edik hatás okoz.

A gátló hatások

A környezetszennyezés

Feltételezhetjük, hogy a környezet pusztítása gátolja a növekedést, mégpedig úgy, hogy csökkenti a születési (pl. genetikai ártalmak) és növeli a halálozási rátát (pl. mérgezések). Mondjuk azt, hogy e kétirányú hatás nagysága csaknem azonos, és függ attól, hogy éppen hány ember él a Földön és szennyezi a környezetet. Ezek alapján:

$$Ksz(1) = Fsz1 * N \quad (4)$$

és

$$Kh(1) = Fh1 * N \quad (5)$$

ahol Fsz1 és Fh1 egy aktuális állandó.

Az energia- és nyersanyagforrások kimerülése

Itt is feltételezhetjük, hogy közvetve ugyan, de mind a születési rátára — csökkenti —, mind pedig a halálozási rátára — növeli — hat. A születési rátára gyakorolt hatást — mivel feltételezhetően kisebb — elhanyagoljuk, és feltételezzük, hogy a hatás első közelítésben független a népesség számától. Ezért:

$$Ksz(2) = 0 \quad (6)$$

és

$$Kh(2) = Fh2 \quad (7)$$

ahol Fh2 egy jól megválasztott állandó.

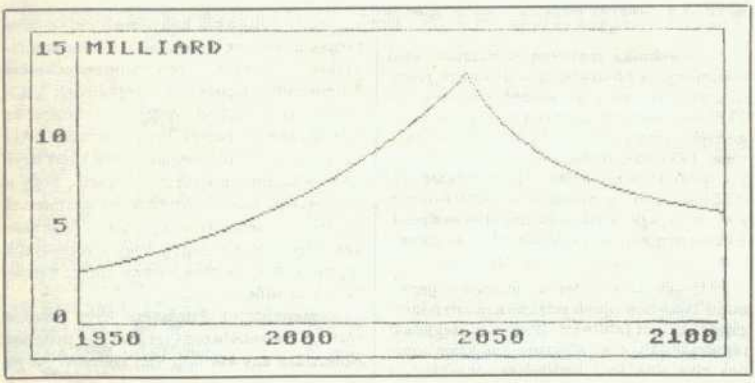
Az élelmiszer-termelés elégtelensége

Itt is érvényes az, hogy bár mindkét tényezőre — születési és halálozási ráta — egyaránt hat, a halálozási rátára gyakorolt hatás jelentősebb, ezért a másikat elhanyagoljuk. Fontosnak tartjuk viszont, hogy a környezetszennyezést itt figyelembe vegyük, és a teljes hatás függjön az aktuális népességszámtól is, tehát:

$$Ksz(3) = 0 \quad (8)$$

és

$$Kh(3) = Fh3 * N \quad (9)$$



4. ábra

valamint

$$Fh3 = Fh3' + Fh3'' \quad (10)$$

ahol Fh3 az aktuális állandó, aminek a környezetszennyezéstől független része Fh3' és függő része pedig Fh3''.

A születésszabályozás

Ez furcsa probléma, a vélemények és ellenvélemények ütközésének legbiztosabb helye. Itt most tételezzük fel, hogy megvalósulhat valamiféle születésszabályozás. Két változatot ajánlunk a modellezőknek. Az első változat az úgynevezett ideális: ez azt jelenti, hogy mindig annyi gyerek születik, ahányan meghalnak; a második a kvázi ideális, vagyis ennél a születési ráta hosszú időn át előre meghatározott — például családonként két gyerek —, a halálozási rátában bekövetkező változásokat viszont nem veszi figyelembe.

Tehát az ideálisnál mindig igaz, hogy:

$$Ksz - Kh = 0 \quad (11)$$

míg a kvázi ideálisnál:

$$Ksz = konst. \quad (12)$$

A természetes vagy mesterséges katasztrófa

Beépíthető a modellbe egy katasztrófahatás is, ami a Föld lélekszámának hirtelen és nagymértékű csökkenésével jár. Ezt a hatást az eddigiektől különválasztva, a népesség pillanatszerű csökkenésével lehet a legegyszerűbben modellezni.

A számítógépes programról

E rövid írásban nem célunk, hogy a modell alapján készült programot közöljük, mert ennek nagy része — adatbeolvasás, grafikus ábrázolás stb. — ismert fogások ismétlése lenne. Az sem elhanyagolható, hogy ha konkrét géptípushoz közölnénk adatokat, akkor a másféle gépek használói joggal elégedetlenkednének. Inkább kedvcsinálónak néhány futtatási eredményt mutatunk be, illetve adjuk meg az egyes paraméterek pontos értékét.

Először bemutatjuk az exponenciális növekedés modelljét *(1. ábra)*. Itt a következő kiindulási paramétereket használtuk az (1) egyenletnek megfelelően: N(0) = 2.5, Ksz = 0.038 és Kh = 0.020.

Most pedig három olyan futtatási eredményt közlünk, amelynél az exponenciális növekedésnek gátakat szabtunk. A kiindulási értékek mindenhol azonosak voltak [N(0) = 2.5, Ksz(0) = 0.038, Kh(0) = 0.020]. Az első esetben *(2. ábra)* az 1980-as évtől vettük figyelembe a környezetszennyezést (Fsz1 = 0.0008 és Fh1 = 0.0014) és (2), (3), (4) és (5) egyenletek szerint.

A második esetben *(3. ábra)* az 1980-as évtől kezdve követtük a környezetszennyezést és 2010-től élelmiszerhiányt is feltételeztünk (Fsz1 = 0.0008, Fh1 = 0.0014 és (Fh3 = 0.0018) a (2), (3), (4), (5), (8) és (9) egyenletek szerint.

A harmadik esetben *(4. ábra)* a 2040-es évben egyszerre jelentkezik a környezetszennyezés, a nyersanyag-, az energia- és az élelmiszerhiány (Fsz1 = 0.0008, Fh1 = 0.0014, Fh2 = 0.005 és Fh3 = 0.0018) a (2), (3), (4), (5), (6), (7), (8) és (9) egyenletek szerint.

Sok sikert a konkrét programok írásához és ezen egyszerű modell továbbfejlesztéséhez, illetve pedagógus kollégáknak az iskolai felhasználáshoz.

DEMETER LÁSZLÓ

Nemes Tihamér

(Budapest, 1895. ápr. 23.—1960. márc. 30.)

A magyarországi televíziózás harmincadik évfordulója kapcsán egyre többet hallunk Nemes Tihamérról, a hazai hírközlés egyik úttörőjéről. Az NJSZT szervezésében lebonyolított Országos Középiskolai Számítástechnikai Verseny is az ő nevét viseli. Nincs ellentmondás. Nemes Tihamér valóban polihisztor volt. Bognár Géza akadémikus a következőket írja többek között Nemes Tihamér Kibernetikai gépek c. könyvének előszavában: „A zseni szerénytelenségével, rendkívül magas igényével veti fel az emberiség legnagyobb kérdését, azt, hogy mi az ember. Erre a kérdésre rendkívül széles műszaki és természettudományi ismeretével, olyannal, amilyennel csak kevesen rendelkeznek, kísérli meg a válaszadást. Az emberi cselekvés és gondolkodás megismerését mérnöki módszerekkel, szerkezeti elemekkel, áramkörökkel közelíti meg. Élete nagy célkitűzése — tudományos munkája során — egyre határozottabban formálódik. A hanganalízisre vonatkozó tanulmányai a beszédíró gépnek képezik részfeladatát. A beszédíró gép szükségszerű logikai funkciói irányították tevékenységét a gépesített agy problémáira. A színes televízióra vonatkozó szabadalmai főleg az emberi látás mechanizmusának mélyebb megismerése szempontjából voltak érdekesek számára. A lépkedő gépre vonatkozó szabadalma az emberi idegrendszer automatizálásának mélyreható tanulmányozásából született. Ezek a nagy jelentőségű felismerések és megoldások a harmincas évekre esnek, egy olyan időszakra, amikor még nem volt ismeretes a kibernetika fogalomköre. Nem voltak ismeretesek azok a módszerek és eszközök, amelyekre a kibernetika tudománya épül. Ebben az időszakban, amikor a tudomány még nem ismerte azt a szót, hogy kibernetika, korát megelőzve kutatja az emberi szervezet és a gépi szerkezet közti közös vonásokat, és ezzel lerakja alapját a kibernetika tudományának."

Nézzük időrendben Nemes Tihamér munkásságát. Oklevelét Budapesten 1917-ben szerezte. 1921-ben a Telefon Hirmondónál, 1929-től a Postakísérleti Állomáson dolgozott. 1950-ben a Távközlési Kutató Intézet tudományos munkatársa lett. 1952-ben a Beloiannisz Híradástechnikai Gyárban dolgozott. 1953-ban a Postakísérleti Állomáson részt vett az első magyar televíziós kép- és hangadó berendezés létrehozásában. Ennek keretében dolgozta ki a 625 soros „flyingsport" filmközvetítő egységet saját találmányú kettős optikai rendszerrel.

Ezután a Magyar Televíziónál dolgozott 1957-ig.

A hőtechnika területén is kutatott, első találmánya a hőszivattyú volt. Ennek jelentőségét azonban csak később ismerték fel. 1930-ban szabadalmaztatta elektronikus orgonáját, amelynek egy oktávját el is készítette. 1933-ban Budapesten a Kossuth Lajos utcai Holczer áruházban rendezett rádiókiállításon bemutatták az általa tervezett és készített távolbalátó berendezést. Ekkor már színes technikával is kísérletezett.

1935-ben találta fel a beszédíró gépet, majd 1944-ben újabb találmányával, a járógéppel keltett feltűnést. 1949-ben megjelent tanulmányában az elektronikus számítógépek elve alapján a kétlépéses sakkfeladványok mechanikus megfejtését elemezte, és közölte gépének elvi rajzát. Később kidolgozott egy logikai gépet, amellyel különböző ok és okozati kapcsolatok automatikusan felismerhetők (v. ö. Magyar Életrajzi Lexikon II. 288. old.).

1957-től a műszaki tudományok doktora. A Budapesti Műszaki Egyetemen oktatott. Könnyen érthető, szemléletes előadásmódjáról volt híres a hallgatóság körében. A Kibernetikai gépek című könyve 1962-ben, halála után jelent meg az Akadémiai Kiadónál. A Rádiótechnika 1960-ban (122. old.) halálára nekrológot közölt, melyben azt írják, hogy Nemes Tihamért kora elismerte a távközlés területén alkotott találmányaiért, de egyéb elgondolásait agya „sziporkázásának" tartották csupán. Példa erre a beszédíró gép, amely a mai kutatások egyik legizgalmasabb területének, a mesterséges értelem kutatásának egyik fő iránya. (Lásd Mesterséges értelem c. sorozatunkat 87/5-ös számunktól kezdődően.)

PINKE GYÖRGY

Méltó körülmények között hirdettek eredményt az 1987. évi Nemes Tihamér Országos Középiskolai Számítástechnikai Tanulmányi Verseny befejezéseként a KISZ KB kongresszusi termében. A szervezőbizottság részéről Zsakó László foglalta össze a verseny tapasztalatait.

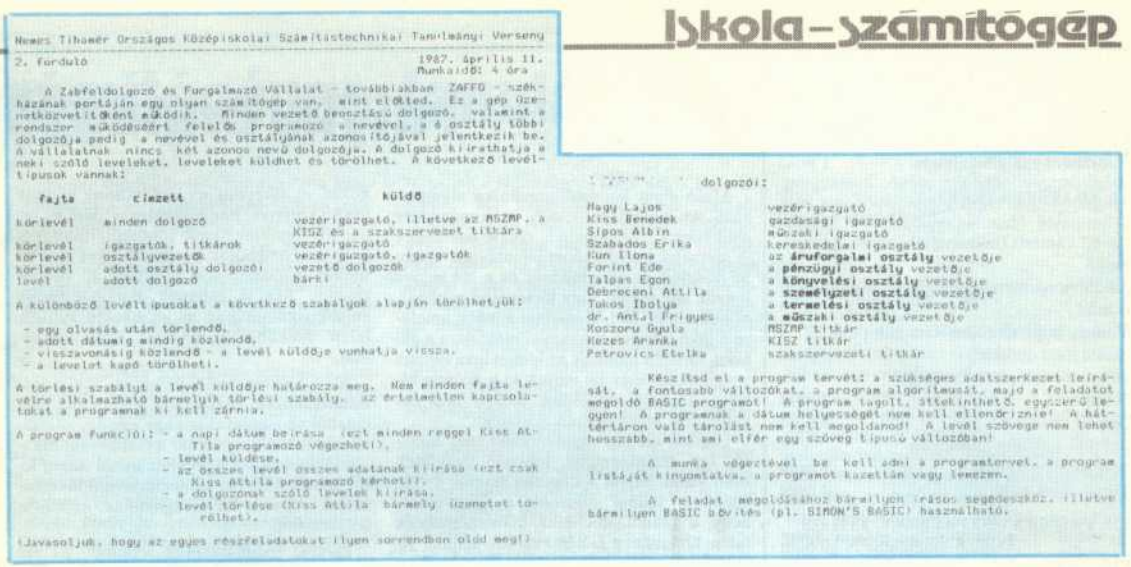
A második forduló feladata egy üzenetközvetítő rendszer programjának megírása volt. Az életből merített probléma jellegét az határozta meg, hogy a dolgozók mágneskártyával kell, hogy bejelentkezzenek a rendszerbe, amely a személyi adataikat tartalmazza. A szervezők nem tudtak a versenyzők számára hálózatot biztosítani, ezért a feladatot egyszerűsítve, egy gépre kellett megoldani.

A versenyen nem született tökéletes megoldás, amit a feladat nagy mérete indokol is, azonban a részfeladatok teljesítését a versenybizottság jól tudta értékelni. A feladat megfogalmazásánál a versenybizottság egyik célja volt, hogy lemérjék, hogyan tudják a tanulók a nagyméretű feladatot részekre bontani, programozástechnikai szempontból kezelni. Az értékelésnél különösen nagy súllyal vették figyelembe az adatábrázolás helyes megválasztását. Aki ezt jól megcsinálta, annak szinte nyert ügye volt. Általános hibaként említhető, hogy a versenyzők nem készítettek programtervet és -leírást, pedig erre felhívták a figyelmüket. Ennek oka az lehet, hogy a versenyzők tapasztalatlanok ezen a téren, illetve sajnálták rá az időt.

A megoldások értékelése nagy munkát rótt a versenybizottságra: egy program pontozása egy-két órát vett igénybe. Az első tizenöt helyezett munkáját kollektíven is értékelték a bizottság kedvéért; és a kedélyek megnyugtatására az eredményhirdetés után a versenyzők megnézhették társaik megoldását.

Az első tíz helyezettnek nem kell matematikából írásbeli felvételi vizsgát tenni az egyetemi, főiskolai jelentkezésnél. Nézzük, kik úszták meg ezt a próbatételt:

1. ERTNER PÉTER
   KOSSUTH LAJOS G.
   NYÍREGYHÁZA — 89 pont
2. DRASNYI GÁBOR
   FAZEKAS MIHÁLY G.
   BUDAPEST — 85 pont
3. RATKAI ISTVÁN
   FAZEKAS MIHÁLY G.
   BUDAPEST — 81 pont
4. RIMÁNYI RICHÁRD
   RÉVAI MIKLÓS G.
   GYŐR — 78 pont
5. KOVÁCS LÁSZLÓ
   I. ISTVÁN G.



Nemes Tihamér Országos Középiskolai Számítástechnikai Tanulmányi Verseny

2. forduló — 1987. április 11. Munkaidő: 4 óra

A Zabfeldolgozó és Forgalmazó Vállalat – továbbiakban ZAFFO – székházának portáján egy olyan számítógép van, mint előttünk. Ez a gép üzemközvetítőként működik. Minden vezető beosztású dolgozó, valamint a rendszer működéséért felelős programozó a nevével, a 4 osztály több dolgozója pedig a nevével és osztályának azonosítójával jelentkezik be. A vállalatnak nincs két azonos nevű dolgozója. A dolgozó kiíratthatja a neki szóló leveleket, leveleket küldhet és törölhet. A következő levéltípusok vannak:

fajta	címzett	küldő
körlevél	minden dolgozó	vezérigazgató, illetve az MSZMP, a KISZ és a szakszervezet titkára
körlevél	igazgatók, titkárok	vezérigazgató
körlevél	osztályvezetők	vezérigazgató, igazgatók
körlevél	adott osztály dolgozói	vezető dolgozók
levél	adott dolgozó	bárki

A különböző levéltípusokat a következő szabályok alapján törölhetjük:

- egy olvasás után törlendő,
- adott dátumig mindig közlendő,
- visszavonásig közlendő – a levél küldője vunhatja vissza,
- a levelet kapó törölheti.

A törlési szabályt a levél küldője határozza meg. Nem minden fajta levélre alkalmazható bármelyik törlési szabály. Az értelmetlen kapcsolatokat a programnak ki kell zárnia.

A program funkciói: – a napi dátum beírása (ezt minden reggel Kiss Attila, Tila programozó végezheti),
– levél küldése,
– az összes levél összes adatának kiírása (ezt csak Kiss Attila programozó kérheti),
– a dolgozónak szóló levelek kiírása,
– levél törlése (Kiss Attila bármely üzenetet törölheti).

(Javasoljuk, hogy az egyes részfeladatokat ilyen sorrendben oldd meg!)

dolgozói:

Nagy Lajos	vezérigazgató
Kiss Benedek	gazdasági igazgató
Sipos Albin	műszaki igazgató
Szabados Erika	kereskedelmi igazgató
Kun Ilona	az **áruforgalmi osztály** vezetője
Forint Ede	a **pénzügyi osztály** vezetője
Talpas Egon	a **könyvelési osztály** vezetője
Debreceni Attila	a **személyzeti osztály** vezetője
Tukos Ibolya	a **termelési osztály** vezetője
dr. Antal Frigyes	a **műszaki osztály** vezetője
Koszoru Gyula	MSZMP titkár
Kezes Aranka	KISZ titkár
Petrovics Etelka	szakszervezeti titkár

Készítsd el a program tervét: a szükséges adatszerkezet leírását, a fontosabb változókat, a program algoritmusát, majd a feladatot megoldó BASIC programot! A program tagolt, áttekinthető, egyszerű legyen! A programnak a dátum helyességét nem kell ellenőriznie! A háttértáron való tárolást nem kell megoldanod! A levél szövege nem lehet hosszabb, mint ami elfér egy szöveg típusú változóban!

A munka végeztével be kell adni a programtervet, a program listáját kinyomtatva, a programot kazettán vagy lemezen.

A feladat megoldásához bármilyen írásos segédeszköz, illetve bármilyen BASIC bővítés (pl. SIMON'S BASIC) használható.

A Nemes Tihamér OKSZTV döntője 1987

	BUDAPEST	77 pont
6.	BOROS PÉTER RADNÓTI MIKLÓS G. SZEGED	75 pont
7.	MAKÓ BALÁZS FÖLDES FERENC G. MISKOLC	75 pont
8.	DUNAY REZSŐ LOVASSY LÁSZLÓ G. VESZPRÉM	74 pont
9.	KUCSMA ISTVÁN IRINYI J. SZ. K. KAZINCBARCIKA	73 pont
10.	MABONY ZSOLT FÖLDES FERENC G. MISKOLC	69 pont

Az NJSZT különdíját a legjobban megtervezett programért PUSKIN ZSOLT, a KISZ KB különdíját pedig az első forduló győztese, SZEPESVÁRI CSABA kapta.

A jövő évi versenytervezet:
első forduló: 1988. január 19.
második forduló: 1988. március 19.

Figyelem! A nevezési határidő 1987. október 31. Cím: NJSZT, 1054 Báthori u. 16.

Olvasóink is próbálkozzanak meg a feladat megoldásával.

Az értékelési szempontok a következők:

I. VEZÉRLŐ RÉSZ ÖSSZESEN 11 pont

BEJELENTKEZÉS 3 pont
— Nem létező vezető beengedése — 1 pont
— Nem létező osztály beengedése — 1 pont
— Nem fogad minden személyt — 1 pont
működik — 1 pont
Látni kell: Címzett, küldő, tartalom
Törlési fajta 2 pont

MENÜ VAGY KÉRDÉSEK SOROZATA 5 pont
— Minden funkció használható 3 pont
— Ugyanaz-e Kissnek Attilának, ha nem 2 pont
— Beenged illegális funkcióba — 1 pont
— Lehet bejelentkezés nélkül bármit tenni — 1 pont

DÁTUMBEÍRÁS 3 pont
— Létezik — 1 pont
— Ha beírása előtt nem lehet semmit csinálni, akkor 2 pont

II. LEVÉL KÜLDÉSE
ÖSSZESEN 17 pont

Csak olyannak küldhet, akinek szabad 4
Ellenőriz-e törlési kapcsolatokat 2 pont
Létezik mindegyik fajta levél 5 pont
Létezik mindegyik törlési mód 4 pont
Küldhet egymás után több levelet 1 pont
Ki lehet lépni küldés nélkül 1 pont

III. ÖSSZES LEVÉL KIÍRÁSA
ÖSSZESEN 4 pont

Levelenkénti lapokra tagolás 2 pont
— Ha nincs levél, rosszul

IV. A DOLGOZÓNAK SZÓLÓ LEVELEK KIÍRÁSA ÖSSZESEN 12 pont

Levelenkénti lapokra tagolás 2 pont
Megkapja a személyesen neki szólókat 2 pont
Megkapja az osztályának szólókat 2 pont
Megkapja a vele azonos beosztásúaknak szólókat 2 pont
Megkapja a mindenkinek szólókat 2 pont
Megtudja, hogy kitől kapta a levelet 2 pont
— ha nem konkrétan — 1 pont
Ha másnak szóló levelet is kap — 4 pont

V. LEVÉL TÖRLÉSE ÖSSZESEN 16 pont

Egy olvasás után törlődik-e 2 pont
Adott dátumkor törlődik-e 3 pont
— Dátumegyenlőség — 1 pont
A küldő visszavonhatja 3 pont
— Ha más is visszavonhatja — 1 pont
— Ha a sikertelen törlésről nincs visszajelzés — 1 pont
— Ha másfajta is lehet — 1 pont
A kapó törölheti 2 pont
— Ha a sikertelen törlésről nincs visszajelzés — 1 pont
Kiss Attila törölhet-e bármit 2 pont
— Csak amit kap — 1 pont
A törlő megtudhatja-e, hogy miből vonhat vissza 2 pont

Ki lehet lépni törlés nélkül	1 pont
A törlésről van visszajelzés	1 pont

VI. ADATSZERKEZET ÖSSZESEN 14 pont

Levél: Címzett (kódolva is), küldő (kódolva is), tartalom, törlési mód, dátum, levélfajta (ha címzésből nem derül ki)	6 pont
Dolgozó: név, küldési jog, beosztás	3 pont
Osztály: név, kód	1 pont
Levelek tárolása (lista, mutató, gyors művelet)	2 pont
Menü, mint adat	2 pont

VII. PROGRAMSZERKEZET, STÍLUS ÖSSZESEN 10 pont

Eljárásokra tagolás, modularitás	3 pont
Közös funkció kiemelése egy eljárásba	2 pont
Megjegyzések, a programszöveg tagolása	2 pont
Egyszerűség, világosság	2 pont
Értelmes változónevek	1 pont

VIII. FELHASZNÁLÓI TULAJDONSÁGOK ÖSSZESEN 10 pont

Kérdések szövegezése	1 pont
Hibajelzések szövege, láthatósága	1 pont
Csak olyan funkciót lát a programból, amit használhat is	2 pont
Törlést csak törölhetőre kérdez	1 pont
Felesleges dolgot nem kérdez	1 pont
Képernyőkezelés esztétikussága	1 pont
A lehetőségekből választás egyszerűsége	3 pont

főprogram, levélfajta, törlési mód

- Ha teljes szavakat kell beírni — 1 pont
- Következetes beolvasás hiánya — 1 pont
- Nem szólít fel billentyűlenyomásra — 1 pont

IX. PROGRAMTERV ÖSSZESEN 5 pont

Adatleírás	
— Tömbök	1 pont
— A kódokat magyarázza	3 pont
— A fontos változók kiemelése	1 pont

ÖSSZESEN 99 pont

PINKE GYÖRGY

Fütyül a magya

Múló divat a számítástechnika alkalmazása az oktatásban vagy a konzervatív szemléletet fenyegető szükségszerűség? A személyi számítógépet jelentősége, szerepe alapján az írásvetítővel és a magnóval tegyük egy sorba, vagy történelmi mérföldkőnek tekintsük? Megannyi kérdés, amelyre a jövő minden bizonnyal meghozza a választ. Örökös harcunk az idővel arra késztet bennünket, hogy ne várjuk ki a holnap bizonyossá váló megoldást, hanem már ma keressünk megnyugtató feleletet.

Logikai alapon egzakt válasz nem adható, így emlékezetünk hasonló helyzetek után kutat. Mivel a képzelet merész és zabolátlan, hamar eljutunk a feltevéshez, hogy a számítástechnika alkalmazásának talán Gutenberg találmányával azonos jelentősége van az ismeretek terjedésében (terjesztésében). Forradalmisága mindkettőnek abban áll, hogy egyszerre információátörökítő és kreativitásra serkentő. Miért prognosztizálható a számítógépek tanulásbeli használatának széles körű elterjedése? Mert az alkalmazás mérlegelésénél alapvető emberi tulajdonságokra — a kényelemszeretetre és a legtöbbünkben meglévő versenyszellemre — gondolva, a jóslat az indítékokra épít.

Visszatekintés

A személyi számítógépekkel segített iskolai oktatás kezdetei nálunk a nyolcvanas évek elejére nyúlnak vissza. Szinte minden felsőoktatási intézményben, de nagyon sok általános és középiskolánkban is kialakult egy-egy kis mag érdeklődő tanárokból és diákokból, akik nekiláttak az akkor hozzáférhető személyi számítógépek feltérképezéséhez, megismeréséhez.

Természetes, hogy az első oktatószoftverek a könnyebben algoritmizálható, a reál szakokhoz közelebb álló tantárgyakból — matematika, fizika, mechanika — születtek. Ezek a programok magukon viselték az első alkotásokra jellemző tulajdonságokat, no és nem lehetett elvonatkoztatni az alkalmazott hardver nyújtotta lehetőségektől, vagy mai szóhasználattal a hardver állította korlátoktól. Az első „ujjgyakorlatok" ABC—80-on, ZX81-en készültek. A HT16 és a ZX—Spectrum megjelenéséig az új eszközök felhasználási lehetőségeit még csak a délelőtt hardveres, délután szoftveres mérnökök, technikusok, illetve a programozás iránt érdeklődő műszaki tanárok kutatták. Fordulat akkor következett a személyi számítógép alkalmazhatóságának megítélésében az oktatásban, amikor a humán tárgyak művelői is felfedezték az új eszközt, a benne rejlő különleges adottságokat, és elkészült az első néhány nyelvet tanító szoftver. Bár ütközetekkel járt, és bizonyos összecsapások többször, több helyütt is lejátszódtak, az eredmények híre és hatása mindeddig szűk szakmai körben maradt. Ma is zajlanak kisebb-nagyobb csaták iskoláinkban és az iskolák mellett, felett működő oktatásügyi intézményekben.

Mi a helyzet ma? Van iskola, ahol a számítástechnikai eszközöket a pedellus, míg máshol az iskolatitkár vagy egy-egy szaktanár őrzi. Mint sasmadár a fiókáit. A fizikaszertárban őrzött gépekhez néhol hozzá lehet jutni csellel, az igazgatóra való hivatkozással, viszont-szívességnyújtás kilátásba helyezésével.

Persze, igazságtalanok lennénk, ha azt állítanók, hogy ez mindenütt így van. Példának okáért a Pécs melletti Szentlőrincen vagy a szegedi Tarján IV-es iskolában, a pesti Radnótiban, József Attilában már nem kell a számítógéppel támogatott módszerek hasznosságáról meggyőzni sem a diákot, sem a tanárt.

De mennyi oktatási intézmény is van Magyarországon? Közel ötezer. És hány iskola tud a példabeliekhez hasonló eredményt felmutatni? Jó esetben 60. Az annyi mint . . . 1,2%.

Peremfeltételek

Megvannak-e a személyi és tárgyi feltételei a számítógép mint taneszköz általánossá válásának, a vele hatékonyabbá tehető tanulás (más szemszögből tanítás) rohamos elterjedésének? Ma nincsenek meg. És a feltételek megteremtésének lehetőségei? Itt bontsuk két részre a választ.

Nézzük először a *személyi oldalt*. Ma tanító pedagógusainknak általában csak a posztgraduális képzés nyújthat lehetőséget azoknak a — ki nem kerülhető — lexikális és gyakorlati ismereteknek a megszerzésére, amelyek nélkül nagy többségüknek nehézségekbe ütközne a személyi számítógépek rendszeres és aktív alkalmazása a tanórákon.

Közel 200 ezer pedagógust érint a történelmi kihívás. Tizenhárom év múlva új évezred küszöbére érkezünk. A ma gyakorló tanárokból akkorra várhatóan kb. 130 ezren lesznek még aktívak vagy a nyugdíj mellett tanítással foglalkozók. Könnyen beláthatjuk, hogy 2001-ben nehezen létezhet majd valaki úgy a pályán, hogy miközben a XXI. században felnövő nemzedéket oktatja, önmaga mindaddig kirekesztetett a minimális számítástechnikai ismereteket nyújtó továbbképzésekből.

számítógépes tanulás vonata

Persze önképzéssel is behozható a lemaradás, de a megszerzendő specifikus ismeretek miatt ez bizonyára nem vezet célra tizezrek esetében. Holott, ha semmi más kedvező hatással nem számolunk — visszatérve prognosztizált számadataink megbízhatóságára —, évente 10 ezer azon pedagógusainknak a száma, akiknek alapképzésben kellene részesülniük.

Hogy milyen időtartamú lehetne a konzultáció? Ma még 5-10 napot elegendőnek tartanak a szakemberek . . .

A számolgatás ráadásul csak akkor helyes, ha feltételezzük, hogy a pedagógusi szakképesítést nyújtó főiskolákon, egyetemeken 1987-től valamennyi végzős — nappalin, levelezőn, estin egyaránt — vizsgakötelesen megismerkedhetett a számítógépes oktatáshoz szükséges tudnivalókkal. Ez — az érintettek jól tudják — ma még nincs így. Inkább az a jellemző, hogy TDK-n, szakkollégiumon, fakultáción lehet összecsipegetni — nem igazán hatékonyan — annyi ismeretet, amely elegendő bátorságot ad a diploma megszerzése után ahhoz, hogy kikerülve az iskolába, a még mindig jobbító szándékú pedagógus elkezdhesse a kísérleteket. E téren tehát mesterségbeli tudásról egyelőre csak jóindulattal beszélhetünk.

Tekintsük át röviden a *tárgyi feltételeket!* Előljáróban nézzük csak meg, zömmel milyen gépek is találhatók a honi iskoláinkban? Mennyi a számuk? Csak felsorolásszerűen: HT16, HT64, Sinclair ZX—Spectrum, Primo, C16, C—Plus/4, C64, VT16, TV—Computer. Bizonyára más típus is előfordul. A mennyiségi becslés tizenötezren áll. Mi ma még csak a technikai minimumra gondoltunk: alapgépre, magnóra, monitorra, no és természetesen szoftverre.

Vezérelt hálózatról, számítógép-video összekapcsolásáról, beszéddigitalizáló berendezések együttes, egyazon rendszerbeli alkalmazásáról nem beszélünk, holott ennek a technikai kivitelezhetősége 1987-ben Magyarországon is fennáll.

Egy-egy iskolán belül az egyes típusok összetétele a legváltozatosabb képet mutatja. Létezik olyan iskola, amelyiknek hét számítógépe van, de öt különböző típusból. Akadnak olyan tanintézmények, ahol képesek egy egész osztályt 20-25 db azonos típusú gép elé ültetni. Erre a nálunk forgalomban levő Commodore család mindegyik tagjával ismerünk példát.

Az iskolák többsége e tekintetben is nehéz helyzetben van. Az egységes géppark kialakítása ma még álom mind az oktatásszervezők, mind a szoftverkészítők szempontjából. Azok az iskolák, ahová úgy került csak a számítógép, hogy nekik (is) osztottak, azt fogadták el, amit kaptak. A szponzorral rendelkező oktatási intézmények kétszeresen vannak előnyös helyzetben: először az e célra fordítható nagyobb összeg miatt, másodszor: a szponzoráló intézmény a kiválasztáskor szakmai segítséget tudott nyújtani (akkor, amikor a számítógép-telepítési tapasztalatban gyenge iskolának erre volt a legnagyobb szüksége).

A gépellátottság tehát jelenleg olyan, amilyen. Az iskolák ma már törekszenek rá, hogy tehetségüktől függően felszereljenek gépekkel egy vagy két labort, amely egyszerre egy-egy osztályt tud kiszolgálni. Ez iskolánként 20—40 gépet jelent; ma 2,5—7 gép az átlag. Nem merész ábránd ez egy kicsit?

1975-ben a négy alapműveletnél és egy gyökvonásnál alig többet tudó zsebszámológép 6000 forintba került. Ma egy hasonló képességű kalkulátort 400 forintért vesztegetnek. Akkor 6000 forintért 1500 liter benzint adtak, ma 400 forintért 20 litert. Az arány meghökkentő: 1/75.

Az eddigiekből úgy tűnhet, hogy a személyi számítógépeknek a tanításbeli/tanulásbeli alkalmazása vagy nem alkalmazása az iskolák belügye, s a tanulónak majdhogynem magánügye. Mondják — és jogosan mondhatják — erre sokan a már közhellyé koptatott igazságot: az iskola, az iskolázás közügy.

Egyszerű lenne most e kijelentéssel zárni a gondolatot: quod erat demonstrandum — ezt kellett bebizonyítani. De miként az oktatás—tanulás nem ér véget az iskola falai között, akként a gépek összeszámlálását is folytatni kell a falakon kívül, kiváltképp a háztartásokban.

1986 végéig az állampolgárok nevére vámkezelt és csak becsülhető magánimport révén több mint 90 ezer személyi számítógép került be az országba: Sinclair ZX—Spectrum és Commodore típusokból van a legtöbb. Az összmennyiség hatszorosa az iskolák birtokában levőnek. A vámcédulás C64-ből és C128-ból állami vállalatok jelentős darabszámban vásároltak, de még így is a nagyobb hányad magánkézben maradt. Majdnem minden géphez ingyen adtak egy vagy két játékkazettát és — többnyire a szülők vagy a nagyszülők — vásároltak hozzá még egynéhányat. A két következő hétvégét végigjátszotta a család apraja-nagyja, majd szép sorban ráuntak a játékra. Maradt a következő feltételezés a hasznosításra: majd a család üdvöskéje ír rá programokat. Sajnos, az első még nem készült el . . .

A közelmúltban megjelentek a boltokban az ismeretszerzést/elmélyítést szolgáló programok. Azóta sok számítógép-tulajdonos újra fölfedezte a gépét az e — tanulást segítő — szoftverekkel való élvezetes és hasznos időtöltés kapcsán.

Szoftverellátottság

Irányelveket és tanulmányokat nem lehet betölteni a háttértárakba, attól még a monitoron nem jelenik meg semmi. Napjainkra azonban már kialakult a mérce a tanulást támogató programok minőségére vonatkozólag. A megkívánt jellemzők közé sok minden tartozik: követelmény a program tartalmi szempontból gazdaggá tétele, lehetőleg a géptípus kapacitása felső határának a kihasználása; ugyanazon szemléltetési ötletre egyazon szerzőtől leginkább csak egy program épüljön, és a szerzők kerüljék a sematikus megoldásokat stb. Mint mindenben, a mesterségbeli tudás itt is csak sok gyakorlás után alakul ki, a szoftverfejlesztés pedig ráadásul eszi is az időt. A jól tanító program készítéséhez a sziklaszilárd szaktárgyi ismereteken túl és a géptől elvárható képességekben való tájékozottságon kívül a tanárnak alaposan tisztában kell lennie a logika, pszichológia, idegélettan számos határterületi kérdésével.

Mivel e képességeknek egy személyen belül való együttes megléte sem túl gyakori, s ennek még hangyaszorgalommal is párosulnia kell, feltehetően csökken azon tanárok száma, akiktől rendszeresen számíthatunk az elvárt színvonalon tanító, korrepetáló programokra.

A jövő útja biztosan az ilyen célú programcsomagoké. Ezeknek a kifejlesztése, gyártása, iskolai bevezetése kellő létszámú szakgárda precíz, hivatásszerű munkáját és együttdolgozását igényli. Lehet-e sikerekről beszámolni?

Magyarország rendezte a számítógéppel támogatott nyelvoktatás I. európai szimpóziumát Debrecenben. A tudós tanárok között három magyar előadó is volt. A szimpóziumon elhangzott előadások anyaga már kapható — igaz, angol nyelven — a könyvesboltokban *Linguistics and Methodology in CALL* címmel. Az elmúlt évben hazánk fogadta az orosz nyelv tanárait világkongresszusukon, amelyen 66 országból 2100-an vettek részt. A szekcióüléseken és a bemutatóteremben látott-hallott magyar oktatószoftverek jó híre négy földrészre eljutott.

Legközelebb arról számolunk be, hol és milyen tárgyú szoftvert talál a Kedves Tanuló a boltokban.

Indul tehát itthon is a számítógépes tanulás vonata: TESSÉK FELSZÁLLNI!

— ij —

Ékezetes karakterkés

A program különlegessége, hogy az ékezetes karaktereket nemcsak a képernyőre, hanem a nyomtatóra is ki tudja írni. Felhasználható bármely BASIC vagy assemblerben írt programban, amelyben ékezetes karakterekre van szükségünk.

A program leírása

A program *(1. lista)* a DATA sorokból beolvasott adatokat tárolja a $COOO-$C1DC memóriaterületre. Az adatok betöltése után a SYS49547 utasítással átmásolja az eredeti karakterkészletet a KARAKTERROM-ból a KERNALROM alatt levő RAM-ba. Ezután a módosított karaktereket másolja át a $COFB címtől kezdve, majd néhány mutató átállításával átvált az ékezetes karakterkészletre.

A program egyszeri lefuttatása után a nyomtatás előtt SYS49152, nyomtatás után SYS49155 utasítással hívhatjuk a gépi kódú rutinokat *(2. lista, 3. lista)*.

A működőképes programot a következőképpen lehet lemezre rögzíteni:

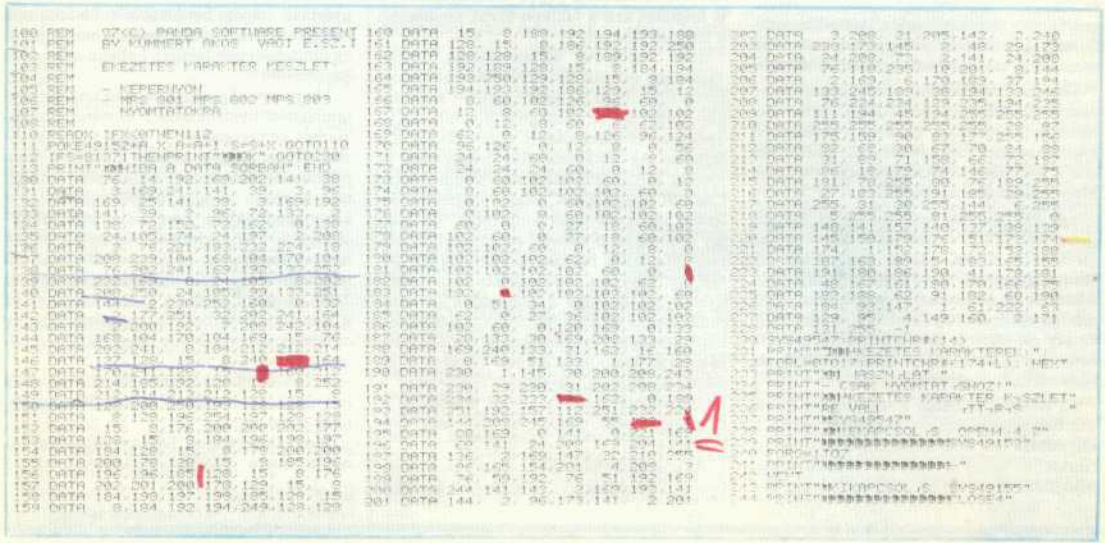
```
10 OPEN1,8,2,"EKES ABC,P,W"

20 PRINT#1,CHR$(O)CHR$(192);

30 FORL=49152TO49840

40 PRINT#1,CHR$(PE-
   EK(L));:NEXT:CLOSE1
```

Ezt a programot figyelmesen gépeljük be! Felhívom a figyelmet a 20. és 40. sorban levő pontosvesszőkre, más a jelentése.

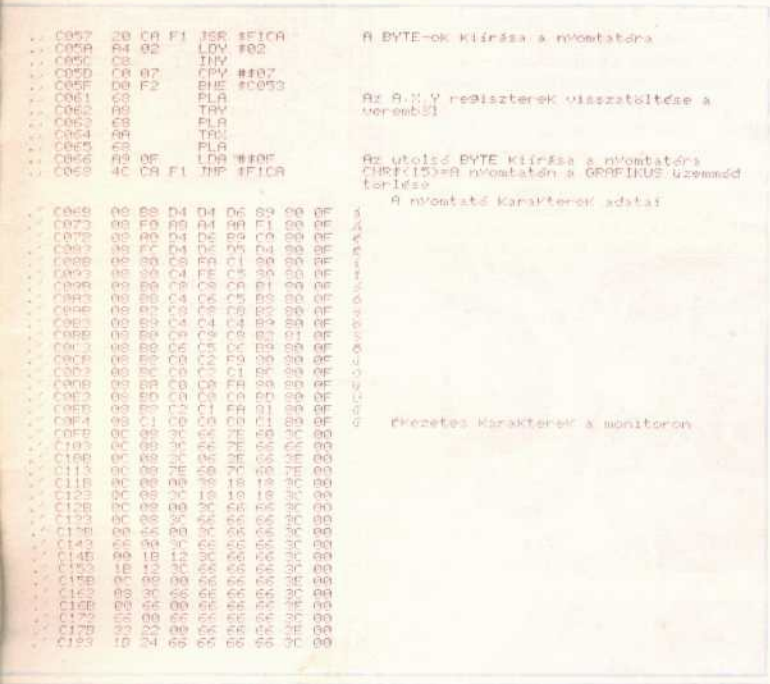
2. lista

```
.; C000  4C 0E C0  JMP $C00E       Nyomtató karakterek beka
.; C003  A9 CA     LDA #$CA        Nyomtató karakterek kika
.; C005  8D 2E 03  STA $032E
.; C008  A9 F1     LDA #$F1
.; C00A  8D 27 03  STA $0327
.; C00D  60        RTS
.; C00E  A9 19     LDA #$19        A PRINT rutin mutatójának a megváltoztatás
.; C010  8D 2E 03  STA $032E
.; C013  A9 CA     LDA #$C0
.; C015  8D 27 03  STA $0327
.; C018  60        RTS
.; C019  48        PHA             Az új PRINT rutin kezdete
.; C01A  95 02     STA $02         Az A,X,Y regiszterek a veremben
.; C01C  8A        TXA
.; C01D  48        PHA
.; C01E  98        TYA
.; C01F  48        PHA
.; C020  A2 00     LDX #$00        Az ékezetes karakter keresése
.; C022  8A        TXA
.; C023  18        CLC
.; C024  69 AE     ADC #$AE
.; C026  18        CLC
.; C027  C5 02     CMP $02
.; C029  D0 03     BNE $C02E
.; C02B  4C DD C1  JMP $C1DD       Ha ékezetes akkor ugrás a $C1DD címre
.; C02E  E8        INX             Ha nem , akkor tovább keresi
.; C02F  E0 12     CPX #$12
.; C031  D0 EF     BNE $C022
.; C033  68        PLA             Ha nem ékezetes karakter , akkor
.; C034  A8        TAY             az A,X,Y regiszterek a veremből.
.; C035  68        PLA
.; C036  AA        TAX
.; C037  68        PLA
.; C038  4C CA F1  JMP $F1CA       Ugrás az eredeti PRINT rutinra
.; C03B  A9 C0     LDA #$C0        Az ékezetes karakterekhez tartozó BYTE -ok
.; C03D  85 FC     STA $FC         helyének kiszámolása
.; C03F  E8        INX
.; C040  A9 00     LDA #$00        C0634*8
.; C042  18        CLC
.; C043  69 08     ADC #$08
.; C045  CA        DEX
.; C046  D0 FA     BNE $C042
.; C048  18        CLC
.; C049  69 63     ADC #$63
.; C04B  85 FB     STA $FB
.; C04D  90 02     BCC $C051
.; C04F  E6 FC     INC $FC
.; C051  A0 00     LDY #$00        A nyolc BYTE beolvasása
.; C053  84 02     STY $02
.; C055  B1 FB     LDA ($FB),Y
```

1. lista



...zlet nyomtatóra



3. lista

sük, mint a PRINT utasításnál. Nem lehet elhagyni!!!

A már rögzített program betöltése vagy a LOAD"EKES ABC",8,1 és utána NEW vagy egy meglévő programból.

```
10 IFA=OTHENA=1:LOAD"EKES
   ABC",8,1

20 IFA=1THENA=2:SYS49547

30 "
   "
```

Az ékezetes betűk billentyűkiosztása

á	SHIFT A
Á	CTRL A
é	SHIFT E
É	CTRL E
í	SHIFT I
Í	CTRL I
ó	SHIFT O
Ó	CTRL O
ő	SHIFT@
Ő	CTRL@
ő	SHIFT*
Ő	CTRL*
ú	SHIFT U
Ú	CTRL U
ü	SHIFT –
Ü	CTRL –
ű	SHIFT £
Ű	CTRL£

KUMMERT ÁKOS

Kétnapi Tízórai

Május 1–3. között rendezte meg a KISZ KB és a Szakmunkástanulók Tanácsa az Országos Diák Napok hagyományos eseménysorozatát. Újdonság volt azonban a Tízórai elnevezésű, nyolcezer példányban megjelent, két számot megért, fényszedéssel készült napilap.

Hét városból — Debrecen, Eger, Gyula, Keszthely, Sárospatak, Sopron, Székesfehérvár — diákújságírók csapata a technika segítségével küldte tudósításait az Ötlet szerkesztőségébe, a budapesti „főhadiszállásra".

A fenti helyszíneken C64-es szövegszerkesztőket használtak, és a lemezre mentett anyagokat a telefonra kapcsolható modem közbeiktatásával juttatták Budapestre. A „központ" az információkat egyszerre négy vonalon fogadhatta, s a tudósításokat városonként külön lemezen tárolta. A kapcsolatot házilagosan kivitelezett, adó- és vevőrészből álló szoftver nyújtotta. Ennek menüje a következő volt: adásra (vételre) állok; adás (vétel) információtárból a modemre, illetve fordítva; töltés (mentés) információ a lemezről a tárba vagy a tárból a lemezre; párbeszéd. Ez utóbbinál az egyik gépen megjelent szöveg a másik — bármelyik városban működő gép — képernyőjén is látható volt.

A „távlapkészítés" folyamatában a begyűjtött anyag újabb gépre került, s az információkat 11 rovat szerinti bontásban lemezre vitték. Egyben az olvasószerkesztőnek kinyomtatták a nyers szöveget, aki átnézte, szelektálta, javította az írásokat, s közben a titkárnők szövegszerkesztőjükön apróbb cikkeket — rovatonként csoportosítva — mentettek lemezre.

A végleges „kézirat" egy C Plus/4-essel került az Ötlet fényszedő rendszerébe, az ottani házi szoftverbe. A szerkesztő kialakította a lap végső formáját, s elkészítette a pozitiv filmet. Ettől kezdve a Szikra Lapnyomdában már hagyományosan montíroztak. A kész újság nevéhez hiven, délelőtt tízre az olvasók kezébe kerülhetett.

ÉNEKES FERENC

DIÁK spectrum ROVAT

Apróságok

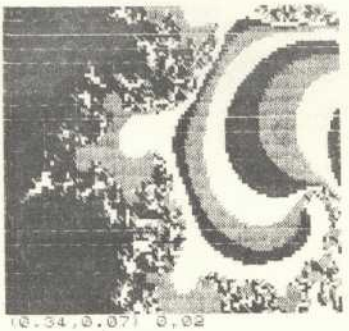
Az első program a ma nagyon népszerű Mandelbrot halmazokról szól. (Ezekről részletesen írtunk a Mikromagazin 1987/1.

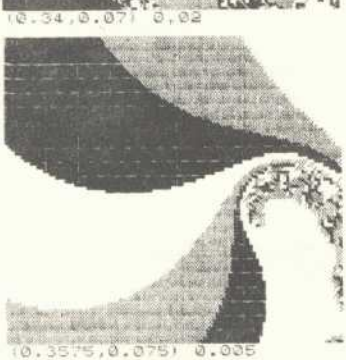
1. program

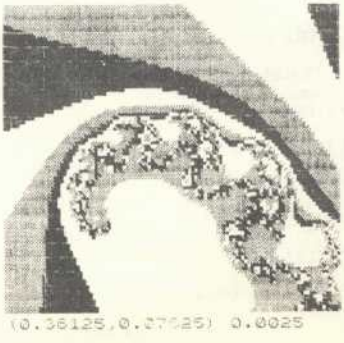
```
  10 PROGRAM MANDELBROT;
  20 CONST R=100;
  30       K=88;
  40
  50 VAR A,B,D,Q,CI,CR,ZI,
  60     ZR,ZR1,Z:REAL;
  70     M,N,X:INTEGER;
  80
  90 PROCEDURE PLOT(X,Y:
 100                 INTEGER);
 110 BEGIN
 120 INLINE(#FD,#21,#3A,#5C,
 130          #DD,#46,#2,
 140          #DD,#4E,#4,
 150          #CD,#E5,#22);
 160 END;
 170
 180 PROCEDURE BEMENET;
 190 BEGIN
 200 PAGE;
 210 WRITE('VALOS ERTEK? ');
 220 READ(A);
 230 WRITE('KEPZETES? ');
 240 READ(B);
 250 WRITE('KEPMERET? ');
 260 READ(D);
 270 Q:=D/K;
 280 END;
 290
 300 PROCEDURE FUGG;
 310 BEGIN
 320   ZR1:=ZR*ZR-ZI*ZI+CR;
 330   ZI:=2*ZR*ZI+CI;
 340   ZR:=ZR1;
 350   Z:=SQRT(ZR*ZR+ZI*ZI);
 360 END;
 370
 380 PROCEDURE FEKETE;
 390 BEGIN
 400 PLOT(M*2,N*2);
 410 PLOT(M*2,N*2+1),
 420 PLOT(M*2+1,N*2);
 430 PLOT(M*2+1,N*2+1);
 440 END;
 450
 460 PROCEDURE PEPI;
 470 BEGIN
 480 PLOT(M*2,N*2);
 490 PLOT(M*2+1,N*2+1);
 500 END;
 510
 520 {FOPROGRAM}
 530 BEGIN
 540 BEMENET;
 550 PAGE;
 560 FOR N:=0 TO K-1 DO
 570 BEGIN
 580 CI:=B+N*Q;
 590 FOR M:=0 TO K+19 DO
 600 BEGIN
 610 CR:=A+M*Q;
 620 X:=0;
 630 ZR:=0;
 640 ZI:=0;
 650 REPEAT
 660 FUGG;
 670 X:=X+1;
 680 UNTIL (X=R) OR (Z>2);
 690 IF X=R THEN FEKETE
 700 IF (X>70) AND (X<100)
 710 THEN PEPI;
 720 IF (X>35) AND (X<50)
 730 THEN FEKETE;
 740 IF (X>25) AND (X<=35)
 750 THEN PEPI;
 760 IF (X>20) AND (X MOD 3=1)
 770 THEN FEKETE;
 780 IF (X>20) AND (X MOD 3=0)
 790 THEN PEPI;
 800 END;
 810 END;
 820 INLINE (#F5,#C5,#D5,#E5,
 830           #CC,#AC,#DE,
 840           #E1,#D1,#C1,#F1);
 850 END.
```

és 2. számában. A szerk.) A Scientific American folyóirat cikke alapján készítettem PASCAL nyelven ZX-Spectrumra.

A második program rövidebb, a magyar szótagolás tanítására, alkalmazására használható. Bárki beépítheti saját programjába. Alkalmazása viszont feltételezi a BASIC nyelv ismeretét. Ez is Spectrumra készült.








```
  5 REM ****Szótagolás******
 10 LET a$="aáeéiíoóöőuúüűAEIOU
    "
 20 LET d$="gytyunytyzsszcs"
 30 INPUT a$
 40 LET w=LEN a$
 50 FOR x=w TO 1 STEP -1
 60 GO SUB 1000
 70 IF p=1 THEN GO TO 100
 80 NEXT x
 90 PRINT a$: LPRINT a$: GO TO
10
100 LET x=x-1
105 IF x=0 THEN GO TO 90
110 GO SUB 1000
120 IF p=1 THEN GO TO 500
125 IF x<2 THEN GO TO 90
130 GO SUB 2000
140 IF p=0 THEN LET x=x-1: GO T
O 500
145 IF x<3 THEN GO TO 90
150 IF a$(x-1) <)a$(x-2) THEN LE
T x=x-2: GO TO 500
160 LET a$=a$( TO x-2)+a$(x)+"-
"+a$(x-1 TO ): LET w=x: GO TO 50
500 LET a$=a$( TO x)+"-"+a$(x+1
TO ): LET w=x: GO TO 50
1000 FOR m=1 TO 19
1010 LET p=a$(x)=m$(m)
1020 IF p=1 THEN RETURN
1030 NEXT m
1040 RETURN
2000 FOR d=1 TO 13 STEP 2
2010 LET p=a$(x-1 TO x)=d$(d TO
d+1)
2020 IF p=1 THEN RETURN
2030 NEXT d
2040 RETURN
```

2. program

A harmadik szintén a Scientific American cikke nyomán íródott BASIC nyelven. A négydimenziós kockákat forgatja.

Valamennyi kipróbálásához jó szórakozást kíván

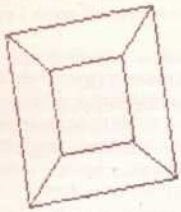
PAPP ZOLTÁN

3. program

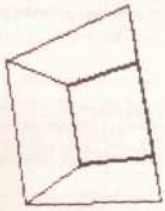
```
  5 LET n=20
  6 LET u=0
 10 DIM c(15,4)
 20 DIM k(16,4)
 30 DIM i(4,4)
 40 DATA 0,1,3,2,5,14,10,8,9,11
,13,7,15,14,12,13,9,11,5,7,6,4,12,
8,6,4,10,15,15,11,10,2,0
100 REM koordsz
110 FOR a=0 TO 1
120 FOR b=0 TO 1
130 FOR c=0 TO 1
140 FOR d=0 TO 1
150 LET u=u+1: LET k(u)=a+b+2*c+d+1
160 LET c(u,1)=(2*d-1)*n
170 LET c(u,2)=(2*c-1)*n
180 LET c(u,3)=(2*b-1)*n
190 LET c(u,4)=(2*a-1)*n
200 NEXT d: NEXT c
210 NEXT b: NEXT a
300 REM forgasmatrix
305 LET u=u+1
310 INPUT "Melyik dim.-bol (1-4
)";d1
320 INPUT "Melyik dim.-ba? ";d2
330 IF d1>d2 THEN GO TO 310
340 INPUT "Szog ";sz
345 LET sz=sz*PI/180
350 FOR i=1 TO 4
360 FOR j=1 TO 4
370 LET r(i,j)=0
380 IF i=j THEN LET r(i,j)=1
390 NEXT j: NEXT i
400 LET r(d1,d1)=COS sz
410 LET r(d2,d2)=COS sz
420 LET r(d1,d2)=-SIN sz
430 LET r(d2,d1)=SIN sz
500 REM szorzas
510 FOR x=1 TO 16
520 FOR y=1 TO 4
530 LET s=0
540 FOR z=1 TO 4
550 LET s=s+c(x,z)*r(y,z)
560 NEXT z
570 LET k(x,y)=s
580 NEXT y: NEXT x
600 REM masolas
610 FOR a=1 TO 16
620 FOR b=1 TO 4
630 LET c(a,b)=k(a,b)
640 NEXT b: NEXT a
700 REM kirajz
702 CLS
705 RESTORE
710 READ l
715 LET l=l+1
720 GO SUB 900
725 PLOT p3+128,p4+88
730 FOR x=1 TO 32
740 LET p1=p3
750 LET p2=p4
760 READ l
765 LET l=l+1
770 GO SUB 900
780 DRAW p3-p1,p4-p2
790 NEXT x
800 PRINT AT 0,0;u;". ";d1;"-";
d2;" elforgatas ";sz*180/PI;" fo
kkal.";
805 COPY
840 GO TO 300
900 REM pont
910 LET p3=c(l,1)*90/(60-c(l,3)
)
920 LET p4=c(l,2)*90/(60-c(l,3)
)
930 RETURN
```

Szintetizátor

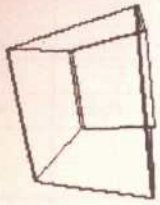
1. 1-2 elforgatas 10 fokkal.



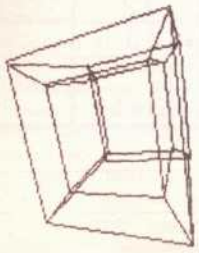
2. 1-3 elforgatas 20 fokkal.



3. 2-3 elforgatas 15 fokkal.



4. 3-4 elforgatas 30 fokkal.



A következő kis gépikód-programmal a Spectrum-tulajdonosok szintetizátor szimulátorként működtethetik gépüket. A program képes a billentyűzeten lejátszott hangok visszajátszására, illetve a visszajátszás sebességének változtatására.

A program két részből, rutinból áll. Az első a billentyűzetről lejátszott zene tárolására, mentésére szolgál. A zene hangjait a 30001 és a 30002 címeken meghatározott területre raktározza. A beírt zene végét ENTER jelzi. Hívása: RANDOMIZE USR 30000. A beírt zenét később felhasználhatjuk, ugyanis kazettán rögzíthetjük. A zene hosszát, a leütött hangjegyek számát a 23296-os cím tárolja, maximális értéke nyilván 255 lehet. Ha pl. 10 billentyűt ütöttünk le, akkor a 23296-os cím értéke 10. Feltéve, hogy a 30001-es és 30002-es címeken 60000 van, a beütött zene kazettára mentése a következő: SAVE "Zene Code" CODE 60000,10.

A rögzített hangok visszajátszása, meghallgatása a következő:

A 31001 éa 31002 címekre beírjuk a felvett zene kezdetét, majd 23296-ba a zene hosszát. Az előző példánál maradva 60000 és 10. Ezután behívjuk kazettáról az első rutinnal kimentett adatokat, majd a RANDOMIZE USR 31000-rel elindítjuk.

Ha a lejátszott zene túl gyors, akkor a 31021,... 31024 címekre 118-at, vagyis a HALT gépi kódú utasítás kódját írjuk be, ami lassítja a lejátszást. Ha a lejátszott zene túl lassú, akkor helyükre 0-t írjunk.

A két rutin tehát egymástól függetlenül is működik, csak a másodiknak van szüksége az elsővel megszerkesztett, kimentett adatokra.

PUSKÁS LÁSZLÓ

1. rutin

```
  10          ORG 30000
  20          ENT 30000
  30          LD BC,60000
  40 NULLA    XOR A
  50          LD (23560),A
  60 VISZA    LD A,(23560)
  70          CP 0
  80          JR Z,VISZA
  90          CP 13
 100          JR Z,RETT
 110          CALL ZENE
 120 BEEP     LD DE,10
 130          PUSH BC
 140          CALL 949
 150          POP BC
 160          JR NULLA
 170 ZENE     SBC A,50
 180          LD (BC),A
 190          INC BC
 200          PUSH BC
 210          LD B,A
 220          LD DE,50
 230          LD HL,0
 240 LOPI1    ADD HL,DE
 250          DJNZ LOPI1
 260          POP BC
 270          RET
 280 RETT     PUSH BC
 290          POP HL
 300          LD DE,60000
 310          SBC HL,DE
 320          LD (23296),HL
 330          RET
```

```
 1 DATA  1, 96,234,175, 50,  0, 92, 58
 2 DATA  0, 92,254,  0, 40,249,254, 13
 3 DATA 40, 30,205, 79,117, 17, 10,  0
 4 DATA 197,205,181,  3,193, 24,228,221
 5 DATA 58,  2,  3,197, 71, 17, 58,  0
 6 DATA 33,  0,  0, 25, 16,253,193,201
 7 DATA 197,225, 17, 96,234,237, 82, 34
 8 DATA  0, 91,201
10 DATA 17, 96,234, 58,  0, 91, 71,197
11 DATA 26,205, 53,121,213,205, 60,121
12 DATA 209, 19,193,110,118,  0,  0,  0
13 DATA  0,118,118, 16,234,217, 33,  0
14 DATA  0, 17, 58,  0, 71, 25, 16,253
15 DATA 209,201, 17, 10,  0,205,181,  3
16 DATA 201
20 FOR F=1 TO 55:READ A:POKE 29999+F,A
21 NEXT F
30 FOR F=1 TO 42:READ A:POKE 30999+F,A
31 NEXT F
```

BASIC betöltő

2. rutin

```
  10          ORG 31000
  20          ENT 31000
  30          LD DE,60000
  40          LD A,(23296)
  50          LD B,A
  60 LOPO1    PUSH BC
  70          LD A,(DE)
  80          CALL VZENE
  90          PUSH DE
 100          CALL BEEP2
 110          POP DE
 120          INC DE
 130          POP BC
 140          HALT
 150          HALT
 160          NOP
 170          NOP
 180          NOP
 190          NOP
 200          HALT
 210          DJNZ LOPO1
 220 VZENE    PUSH DE
 230          LD HL,0
 240          LD DE,50
 250          LD B,A
 260 LOPO2    ADD HL,DE
 270          DJNZ LOPO2
 280          POP DE
 290          RET
 300 BEEP2    LD DE,10
 310          CALL 949
 320          RET
```

OKTA-TOTÓ

Megfejtés

— A rendszerprogramoknak közvetlenül semmi közük a felhasználói programokhoz, azok szintaktikus hibáit csak a fordítóprogram észleli (2).

— A Neumann-elv egyik sarkalatos pontja, hogy a programok a futásuk során hajtják végre az utasításokat (1).

— A beviteli-kiviteli utasítások (illetve általában az utasítások) sebessége nem függhet felhasználói programtól. A processzor sebessége pedig mindig nagyobb, mint a perifériáké (2).

— A DOS a Disc Operating System (lemezes operációs rendszer) rövidítése (X).

— A BASIC interpreter rendszerint egy bemenő fizikai blokkot tekint egyszerre át, vagyis ha egy sorban több utasítás van, akkor is az egész sort (nem külön az utasításokat) (2).

— A FORTRAN a FORmula TRANslator (képletátalakító) rövidítése, tehát inkább matematikaorientált programnyelv (X).

— A magas szintű programnyelvek előnye a könnyebb kezelhetőség, az egyszerűbb programírás és -javítás stb. Kifejezetten hátrányuk, hogy a tárkihasználásuk nem a legjobb, és a futási idő is nő (2).

— A vezérlésátadó utasítások természetesen a programon belül vagy egy másik utasításnak, vagy egy tetszőleges másik tárcímre adják a vezérlést (1).

— A deklarációk nem mindig kötelezőek; a használatos változóknak foglalnak helyet (2).

— Ha a futás adathiba miatt szakad meg, akkor nyilván az adatokat kell először ellenőrizni és szükség esetén javítani (1).

— Egy program nem akkor optimális, ha a legkevesebb sorban fér el (hiszen akkor például szinte egyáltalán nem módosítható), és ha sok GOTO utasítást tartalmaz, akkor áttekinthetetlen. A DO (ciklus)utasítások jobban szervezetté teszik a programokat (X).

— A BASIC-ben szubrutinból térhetünk vissza RETURN-nel (X).

— Valószínűleg olyan program egyáltalán nem létezik, amelyet többféle fordítóprogrammal is le lehetne fordítani, vagy a működése független lenne a bemenő adatoktól (1).

— A szerkesztő-összefűző programok olyan rendszerprogramok, amelyek futtatható programot – ún. fázist – állítanak elő (2).

A helyes megfejtés tehát összefoglalva:

1 2 3 4 5 6 7 8 9 10 11 12 13 13+1
2 1 2 X 2 X 2 1 2 1 X X 1 2

---

1. Az IBM személyi számítógépeiben milyen mikroprocesszort alkalmaznak?
   1. Motorola
   2. National Semiconductor
   X. Intel

2. Mi az alapvető különbség az IBM PC sorozatának itthon is elterjedt elemei között?
   1. a használatos monitor /színes vagy fekete-fehér/
   2. a perifériakészlet
   X. az órajel sebessége

3. A Commodore gyárnak melyik az IBM PC-val kompatibilis gépe?
   1. Amiga
   2. C-8000
   X. PC-20

4. Melyik az IBM PC-k legsikeresebb felhasználói programnyelve?
   1. BASIC
   2. PASCAL
   X. FORTRAN

5. A "professzionális" célú programoknak a személyi számítógépeken belül kb. hány százaléka készült IBM PC sorozatra?
   1. 65
   2. 40
   X. 15

6. Az IBM személyi számítógépek melyik típusa képes a legnagyobb fizikai tárat kezelni?
   1. PC
   2. PC XT
   X. PC AT

7. Egy IBM PC AT konfigurációban lehet
   1. 1,2 Mbájt hajlékonylemez
   2. 40 Mbájt merevlemez /Winchester/
   X. mindkettő

8. Mi az az NLQ?
   1. nyomtaták jellemzője
   2. memóriaméret-jellemző
   X. háttértároló-kapacitás jellemző

9. Mi az a streamer?
   1. A merevlemezek tartalmának teljes vagy részleges kimentésére szolgál
   2. hagyományos kazettás mágnesszalag
   X. mágneslemez

10. Mi a legnagyobb különbség az eredeti IBM és az utánzat személyi számítógépek között?
    1. a sebesség
    2. a programellátottság
    X. a megbízhatóság

11. Az IBM PC sorozatának kompatibilitása miatt
    1. az XT-futó programok az AT-n is futnak
    2. az AT-n futó programok bármely másik gépen is futnak
    X. a sorozaton belül bármely program bármelyik gépen fut

12. Melyik a tipikus IBM PC szoftver?
    1. Framework
    2. dBASE III
    X. mindkettő

13. Mi az a "directory" /katalógus/?
    1. a lemezen lévő állományokat tartalmazza
    2. a lemezen lévő állományok neveit tartalmazza
    X. a lemezen lévő adatokat tartalmazza

+1. Az alábbi IBM PC utánzatok közül melyik a leggyorsabb?
    1. Olivetti PC XT
    2. 420 XT
    X. Varyter XT

Beküldési határidő: 1987. augusztus 23. Kérjük az olvasókat, hogy a totószelvényt postai levelezőlapra ragasztva küldjék be a szerkesztőség címére. (1371 Budapest, Pf.: 433.) Szelvény nélkül érkezett megoldások nem vesznek részt a sorsolásban!

PÁLYÁZATI SZELVÉNY 87/8

OKTA-TOTO

Kérdés	Tipp
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	
11.	
12.	
13.	
+	
13+1.	

Név: _____

Cím: _____

_____

Szem.szám.

Tömb- és fájlkezelés

A számítógép-alkalmazások jelentős részében floppyn tárolt adatokat dolgozunk fel. Feldolgozáskor az adatokkal többféle műveletet kell végeznünk:

— adatbevitel és -tárolás floppyn egy állományban;

— újabb adatok beépítése már meglévő állományba;

— adatok olvasása az állományból;

— az állományban lévő adatok módosítása.

Ha az adatok mennyisége olyan kicsi, hogy a kezelőprogram mellett még elférnek a memóriában, akkor az adatkezelés legcélszerűbb módja a memórián belüli kezelés. Ilyenkor csak azért tároljuk adatainkat floppyn, hogy azok a gép kikapcsolása után is megmaradjanak, és később újra feldolgozhatók legyenek. Ilyen tároláshoz elsősorban kevés szoftverrel rendelkező számítógépeken viszonylag egyszerűbb módszerek kínálkoznak.

Cikkünk célja, hogy megismertessük az olvasót az ATARI—800XL és az 1050-es floppy állománykezelésének egyes változataival, természetesen ismét párhuzamba állítva a Commodore 64/VIC—1541 összeállítás hasonló képességeivel. Az állománykezelés közben felvetünk néhány problémát az A—800 karakterkezelésével kapcsolatban, mert ez jelentősen eltér a C64 karakterkezelési megoldásaitól.

Mivel mindkét gép „alapnyelve" a BASIC, itt csak a BASIC-ből könnyen kezelhető állományokkal foglalkozunk. Az állományok kezelésére a C64-en látszólag többféle lehetőség van a DOS teljes felügyelete alatt, mint az A—800XL-en. Ismerv – vagy több jellemző – szerint kell az adatokat visszakeresni.

Ezért a C64-en is vagy valamilyen „saját" fájlkezelő rendszert kell kialakítani, vagy valamelyik általános adatkezelő szoftvert kell megvásárolni. Az A—800-as BASIC-ből „csak" a szekvenciális állományszervezési lehetőség áll rendelkezésünkre. Van azonban két BASIC utasítás, mely az állománykezelési lehetőségeket kiterjeszti, közvetlen elérést teremt. A NOTE utasítás, amely egy rekord relatív helyét adja meg egy fájlban, és a POINT utasítás, amellyel a következő művelet előtt a rekordmutatót lehet pozicionálni. Mindkét utasításban a rekordmutató két változóban helyezkedik el. Az első a relatív szektorszámot, a második a szektoron belüli bájtpozíciót adja meg.

ADATALLOMANY

CIM		TELEFONSZAM	ELŐFIZETŐ NEVE
s	b	6	24
0	0	692733	JO CSABA
0	30	434011	SZéP IZABELLA
0	60	149668	KISS LORANT
0	90	845610	NAGY ISTVAN
0	120	334937	SZABO TIBOR

TELEFONSZAM INDEXTABLA

TELEFONSZAM	s	b
149668	0	60
334937	0	120
434011	0	30
692733	0	0
845610	0	90

NéV INDEXTÁBLA

ELŐFIZETŐ NEVE	s	b
JO CSABA	0	0
KISS LORANT	0	60
NAGY ISTVAN	0	90
SZABO TIBOR	0	120
SZéP IZABELLA	0	30

gendő a szekvenciális tárolási és hozzáférési módszer. Ez azt jelenti, hogy az adatokat sorban egymás után visszük ki az adathordozóra, és csak ugyanilyen sorrendben olvashatjuk vissza.

Ha nagyobb mennyiségű adattal dolgozunk, akkor nemcsak tárolni kell a floppyn, hanem a feldolgozást is úgy kell szervezünk, hogy a tárolt állomány adatai egyszerűen elérhetők legyenek. Bizonyos esetekben természetesen így is beválnak a szekvenciális állományok, ilyenkor azonban az egyes adategységek elérése és feldolgozása nagyon időigényes lehet, mert az összes megelőző adatot is be kell olvasni a floppyról. Ezért minden gépnél, amelyen nagy tömegű adatot akarunk feldolgozni, igen fontos, hogy legyen lehetőség az adatok úgynevezett közvetlen elérésére. Ehhez a kimert például a szekvenciális állományszervezés (SEQ), és az így szervezett állomány adatai BASIC-ből valóban csak sorban egymás után olvashatók.

A C64 másfajta elérést biztosító állománytípusát relatívnak (REL) hívják. Ennél az állományszervezésnél az egyes adategységek (adatmondatok) az állományban elfoglalt helyük sorszáma alapján olvashatók. Az egyes adatmondatok felvitelekor kell kijelölni a helyüket, és azontúl ezen a helyen találhatók meg.

Az egyes adatmondatok rögzített hosszúságúak. A DOS a sorszám és a mondathossz alapján tudja meghatározni a pontos sáv, szektor, illetve az azon belüli bájt címét. A sorszám azonban nagyon kevés kivételtől eltekintve nem jó azonosító, valamilyen egyéb, az adatmondatot meghatározó ismérv – vagy több jellemző – szerint kell az adatokat visszakeresni.

Az A—800XL ezzel a két utasítással módot nyújt arra, hogy egy rekord helyét lemezre írása után feljegyezzük. Ha ugyanakkor megjegyezzük a rekord azonosítóját vagy az adatmondat általunk választott jellemzőit, akkor később a rekordot közvetlenül ennek alapján tudjuk elérni.

Ha ezeket a „feljegyzéseket" és a hozzájuk tartozó keresési ismérveket táblázatba foglaljuk, akkor egy teljes indextáblát kapunk. Ha az indextáblát egy másik (index) állományban tároljuk is, akkor indexelt állományt kapunk.

Természetesen mind a keresés, mind az egyéb kezelési műveletek gyorsabbak, ha az indextáblát az egyes keresési jellemzők alapján növekvő (vagy csökkenő) sorrendbe állítjuk. Ez olyankor is szükséges lehet, ha a különböző rendezettségi szempontok

(kulcsok) alapján az állományból listát (sorrendbe állított rekordok sorozatát) akarunk készíteni (például névsorrendbe állított telefon-előfizetők).

A rendezett indextábla tehát további előny; ezt a szerkezetet nevezik index-szekvenciális állománynak. Az *ábrában* erre láthatunk egy kis példát.

Egyelőre csak nagyon egyszerű kis programokat, modulszerűen mutatunk be. Ebből már felépíthetők tetszőlegesen bonyolult programok is, és a következő részben erre is hozunk majd példát.

Az első egyszerű adatállomány mondatai mindössze két jellemzőből állnak:

— egy 6 karakter hosszúságú sztring a telefonszámot (T$),

— egy 24 karaktert tartalmazó sztring az előfizető nevét tartalmazza (N$).

Az első példa a fájl létrehozását mutatja be. A program *(1. lista)* mindaddig kéri a billentyűzetről a telefonszámot és a nevet, amíg a telefonszám helyett egy „*" karaktert nem viszünk be. Részletezve a programot:

20 megnyitja #1 logikai számmal új fájlként az 1-es (alapértelmezés szerinti) meghajtón a DATA.DAT nevű adatállományt. A második paraméterben a 8 azt jelenti, hogy új fájlt hozunk létre, amibe csak írni lehet;

30 megnyitja #2 logikai számmal új fájlként az INDEX.NDX nevű állományt (szintén csak írás);

32 sztringek hosszának meghatározása;

40 képernyőtörlés;

50—60 adatbevitel;

70 végfeltétel-vizsgálat;

80 kiírás az adatfájlba;

90 rekordmutató lekérdezése;

100 indexrekord felvitele az indexfájlba. A keresési ismérv (kulcs) a telefonszám;

110 visszatérés újabb adat olvasására;

1000—1020 a munka végén a fájlok zárása és programbefejezés.

A második program *(2. lista)* példa a meglévő állomány bővítésére. Különbség az előzőhöz képest a megnyitás módjában van a 20-as és a 30-as sorokban. Az OPEN második paramétere 9, ami azt jelenti, hogy egy meglévő állomány végére írunk.

A harmadik program *(3. lista)* példa egy indexelt állomány közvetlen elérésére a kulcs alapján. Részletesen (csak azok az utasítások, amelyek az eddigiekből nem ismertek):

30—31 már ismert megnyitások azzal a különbséggel, hogy az OPEN második paramétere 4, vagyis a fájlokból csak olvasni lehet;

32 billentyűzet megnyitása;

38 karaktersztringek kijelölése; X$ a keresett telefonszám számára kijelölt hely;

50 a keresett telefonszám bekérése;

60 indexfájl olvasása: telefonszám, szektorszám, bájtpozíció;

65 hiba (fájl vége) esetén 300-ra ugrik, ahol kiírja, hogy nincs ilyen szám (nem szerepel az indexfájlban);

70 összehasonlítja a keresett telefonszámot az indexfájlból olvasottal;

80—100 pozicionál és olvas a mutató alapján, kiírja képernyőre a telefonszámot, a nevet és a mutatókat;

110 az indexfájl mutatóját a fájl elejére állítja, hogy a következő keresést onnan lehessen kezdeni.

Az indextáblák az esetek nagy részében tárolhatók a memóriában is, ami a keresést és az elérést nagyon meggyorsítja. Ennek módszere, hogy az indexfájlt egyszer, a program elején beolvassuk. A kulcsokat egy karakteres tömbbe, a mutatókat pedig egy-egy numerikus tömbbe helyezzük el.

```
10 REM **********************
11 REM * INDEXELT FAJL IRAS *
12 REM **********************
20 OPEN #1,8,0,"D:DATA.DAT"
30 OPEN #2,8,0,"D:INDEX.NDX"
32 DIM T$(6),N$(24)
40 GRAPHICS 16
50 PRINT "TELEFONSZAM,NEV"
60 INPUT T$,N$
70 IF T$="*" THEN 1000
80 PRINT #1;T$,N$
90 NOTE #1,S,B
100 PRINT #2;T$;S;B
200 GOTO 40
1000 CLOSE #1
1010 CLOSE #2
1020 END
```

1. lista

Az A—800XL-nél lényegében nincs mód a karakteres tömbkezelésre, a dimenzionálás csak a sztring elemeinek számát határozza meg. Ezért a szakaszolást (sztringtömb kialakítását) a programban a sztringindex algoritmusa biztosítja. A tömbméretek meghatározására a negyedik példában *(4. lista)* feltételezzük, hogy maximum 20 elemű az indextábla, így a telefonszámok tárolására 120 karakter szükséges. A program részletes működése:

38 TS$ = telefonszám
N$ = név
X$ = keresett telefonszám
T$ = telefonszámtömb (az indextábla kulcsai)
P, Q = mutatótömbök (az indextábla mutatói)
W$ = sztring segédtároló

50—72 indextábla beolvasása ciklikusan. A sztringtömböt 6 karakterenként szakaszosan töltjük fel;

74 N az indextábla tényleges mérete (< = 20). Egyet azért kell levonni, mert a ciklusból az utolsó elem olvasása után lép ki, illetve normál esetben a ciklust elhagyva a ciklusparaméter utolsó végrehajtott paraméter + növekmény (1) értéket vesz fel;

80—95 a keresés az indextábla kulcsának sztringjében. Ha a keresésből a ciklus végén lép ki, azt jelenti, hogy nincs a táblázatban a keresett szám (95-ös sor). Ha megtaláltuk a keresett számot, a program a 100-as soron folytatódik (84-es sor);

100—200 az ismert módon beolvassuk a keresett rekordot úgy, hogy a mutatótömb megtalált I-edik elemével pozicionálunk (100-as sor). A 140-es sorban a GET várakozást tart fenn mindaddig, míg egy billentyűt meg nem nyomunk.

Ahhoz, hogy az indexelt fájlból indexszekvenciálisat alakítsunk ki és a keresést tovább gyorsítsuk, írásunk további részében előbb foglalkoznunk kell a memórián belüli rendezés és keresés lehetőségeivel.

```
10 REM **********************
11 REM * FILE KEZELES PROBA *
12 REM **********************
20 OPEN #1,9,0,"D:DATA.DAT"
30 OPEN #2,9,0,"D:INDEX.NDX"
32 DIM T$(6),N$(24)
40 GRAPHICS 16
50 PRINT "TELEFONSZAM,NEV"
60 INPUT T$,N$
70 IF T$="*" THEN 1000
80 PRINT #1;T$;N$
90 NOTE #1,S,B
100 PRINT #2;T$;S;B
200 GOTO 40
1000 CLOSE #1
1010 CLOSE #2
1020 END
```

2. lista

A fájlkezelési feladatok gyakorlatban szokásos megoldásaiban elengedhetetlen a karakteres tömbök kezelése. Túl azon, hogy előző cikkünkben igéretet tettünk az A—800XL-en szokatlan formában kezelt sztringtömbök rendezésének, kezelésének bemutatására, a komplex igényű feladatmegoldás is szükségessé teszi, hogy erre most a fájlkezeléssel összefüggésben kerüljön sor.

Ismeretes, hogy az A—800XL a sztringtömböket folytonosan tárolja. A tömb úgy jön létre, hogy az új tömbelemeket folyamatosan hozzákapcsoljuk az előzők együtteséhez. Ésszerű tehát, ha az egyes tömbelemeket állandó hosszúságúra választjuk, mivel így könnyebb lesz a tömb elemenkénti kezelése, mechanikussá válik az elemek elhatárolása.

```
10 REM ************************
11 REM * NDX FILE DIREKT OLV *
12 REM ************************
30 OPEN #1,4,0,"D:DATA.DAT"
31 OPEN #2,4,0,"D:INDEX.NDX"
 OPEN #3,4,0,"K:"
38 DIM T$(6),N$(24),X$(6)
40 GRAPHICS 16
50 PRINT "TELEFONSZAM":INPUT X$
60 INPUT #2,T$,P,0
65 TRAP 300
70 IF T$=X$ THEN 80
72 GOTO 60
80 POINT #1,P,0
90 INPUT #1,T$,N$
100 PRINT T$;":";N$;":";P;"
110 S=0:R=0:POINT #2,S,R
120 GET #3,C
200 GOTO 40
300 PRINT "NINCS ILYEN SZAM!":GOTO 110
1000 CLOSE #1
1010 CLOSE #2
1020 CLOSE #3
1030 END
```

3. lista

hordozott információ egészére, akkor célszerű úgy eljárnunk, hogy minden tömbelemet a tömbbe helyezés előtt üres helyekkel kiegészítünk, és az így megnövelt hosszt vágjuk vissza a fix hosszra.

Lássunk egy példát az utóbbira. Tegyük fel, hogy programunk alkalmazási körében egy tömbelem hossza sem haladja meg a 20 karaktert. Válasszuk tehát az állandó szóhosszt 20 karekterre. Ezek után azt kell biztosítanunk, hogy minden szavunk valóban mindig ilyen hosszú legyen. Legyen az eredeti szavunk B$,

akkor a fix szóhossz beállítása:

B$ (LEN(B$)+1)=" "
B$=B$ (1,20)

A tömbelemek kiválasztására a kiemelő függvényt használjuk, ehhez viszont meg kell feleltetni a tömbelem sorszámát és kezdőkarakterének tömbbeli sorszámát egymásnak. Ez a következőképpen tehető meg:

Legyen a tömbelem sorszáma S. Ha a kezdőkarakter K: K=(S—1)*20+1

Az elem utolsó karaktere pedig (ha a szóhossz továbbra is 20)

U=S*20

Így egy A$ tömb S-edik eleme:

A$(K,U)

Most pedig lássunk példát a tömb helyben rendezésére. Felhasználjuk a fenti összefüggéseket, s a „buborék" algoritmust használjuk *(5. lista)*.

```
10 REM ************************
11 REM * NDX FILE DIREKT OLV *
12 REM ************************
20 CLR
30 OPEN #1,4,0,"D:DATA.DAT"
31 OPEN #2,4,0,"D:INDEX.NDX"
32 OPEN #3,4,0,"K:"
35 CLR
38 DIM TS$(6),N$(24),C$(1),X$(6),T$(128),P(20),0(20),W$(6)
40 GRAPHICS 16
50 FOR I=1 TO 20
60 INPUT #2,TS$,PS,0S
62 T$((I-1)*6+1)=TS$:P(I)=PS:0(I)=0S
64 PRINT T$((I-1)*6+1),P(I),0(I)
68 TRAP 74
72 NEXT I
74 N=I-1:PRINT N
78 PRINT "TELEFONSZAM":INPUT X$
80 FOR I=1 TO N
82 W$=T$((I-1)*6+1)
84 IF W$=X$ THEN 100
90 NEXT I
95 PRINT "NINCS ILYEN SZAM!":GOTO 140
100 POINT #1,P(I),0(I)
110 INPUT #1,TS$,N$
120 PRINT TS$;":";N$;":";P(I);":";0(I)
130 S=0:R=0:POINT #2,S,R
140 GET #3,C
150 GRAPHICS 16
200 GOTO 78
1000 CLOSE #1
1010 CLOSE #2
1020 CLOSE #3
1030 END
```

4. lista

```
5 PRINT "A TOMBELEMEK SZAMA":INPUT N
6 PRINT "A TOMBELEMEK ALLANDO HOSSZA":INPUT H:NH=N*H
10 DIM A$(NH),B$(30)
20 FOR I=1 TO N
30 INPUT B$
35 B$(LEN(B$)+1)="
36 B$=B$(1,H)
40 A$(LEN(A$)+1)=B$
50 NEXT I
60 FOR I=0 TO N-2
65 I1=I*H+1:I2=(I+1)*H
70 J=I+1
80 FOR K=J TO N-1
85 K1=K*H+1:K2=(K+1)*H
90 IF A$(I1,I2)<A$(K1,K2) THEN 110
100 B$=A$(I1,I2):A$(I1,I2)=A$(K1,K2):A$(K1,K2)=B$
110 NEXT K
120 NEXT I
130 PRINT A$
```

5. lista

```
5 PRINT "TOMBELEMEK SZAMA":INPUT N
6 PRINT "TOMBELEMEK ALLANDO HOSSZA":INPUT H:NH=N*H
10 DIM A$(100),B$(30)
20 FOR I=1 TO N
30 INPUT B$
35 B$(LEN(B$)+1)="
36 B$=B$(1,10)
40 A$(LEN(A$)+1)=B$
50 NEXT I
60 FOR J=1 TO N-1
65 I0=(J-1)*H+1:I1=I*H:I2=(I+1)*H
70 B$=A$(I1+1,I2)
80 FOR I=I TO 1 STEP -1
85 I0=(I-1)*H+1:I1=I*H:I2=(I+1)*H
90 IF B$>A$(I0,I1) THEN 130
100 A$(I1+1,I2)=A$(I0,I1)
110 NEXT I
120 I1=0:I2=H
130 A$(I1+1,I2)=B$
140 NEXT I
150 ? A$
```

6. lista

```
1 PRINT "AZ ADATOK SZAMAT ES HOSSZAT KEREM":INPUT N,H:NH=N*H
2 DIM A$(NH),B$(30)
3 FOR I=1 TO N
4 INPUT B$:B$(LEN(B$)+1)="
5 B$=B$(1,H)
6 A$(LEN(A$)+1)=B$
7 NEXT I
10 B$="":PRINT "A KERESETT SZO";
15 INPUT B$:B$(LEN(B$)+1)="
16 B$=B$(1,H)
20 L=INT(LOG(N)/LOG(2)+1)
30 AH=1
40 FH=N+1
50 K=INT((AH+FH)/2)
60 KA=(K-1)*H+1:K1=K*H
70 IF A$(KA,K1)=B$ THEN PRINT A$(KA,K1):GOTO 10
80 IF A$(KA,K1)>B$ THEN FH=K:GOTO 100
90 AH=K
100 L=L-1:IF L<0 THEN PRINT "NINCS ILYEN":GOTO 10
110 GOTO 50
```

7. lista

Ezt úgy érjük el, hogy vagy a hosszabb tömbelemeket egy alkalmas hosszra vágjuk, vagy ha szükség van a tömbelem által

Az így beállított szót kapcsoljuk a tömbhöz. Ha a tömb A$:A$(LEN(A$)+1)==B$. Ez egy könnyen kezelhető tömb.

Kisebb tömbök rendezésére ez jól áttekinthető algoritmus, viszont lassú. Kielégítően gyors, de még mindig könnyen programozható az ún. beszúró (INSERT) rendezés *(6. lista)*. A rendezett tömbben igen gyorsan tudunk keresni.

A *7. listán* bemutatott eljárás az ún. bináris keresés. Alapgondolata, hogy a tömb rendezettsége következtében folyamatos felezéssel mindig csak a maradék halmazt kell vizsgálni.

Következő cikkünkben a fentiek általános alkalmazását fogjuk bemutatni.

JÁNOSA— PAÁL—SÜTŐ

BASIC és gépi kód

Legutóbb olyan programokat láttunk, melyek csillagokkal írják tele a Commodore gépek képernyőjét, és összehasonlítottuk a sebességüket. Most — egy újabb utasításcsoporttal való ismerkedés után — megnézzük a múltkori összehasonlításban leggyorsabb programot.

A léptetőutasítások

A léptető (shiftelő) utasítások az A regiszternek vagy az operandus által meghatározott memóriabájtnak a tartalmát léptetik egy bitpozícióval jobbra vagy balra. Négy utasítás tartozik ebbe a csoportba. Működésük vázlatát a segédlet JELÖLÉSEK című részének utolsó négy sorában láthatjuk, a következő sorrendben: LSR, ASL, ROR, ROL.

A léptetőutasítások a Z, N és C állapotbiteket állítják be. A Z és N bit tartalma — a korábban megismert utasításokhoz hasonlóan — az operandusnak az utasítás végrehajtása utáni tartalmától függ. N-be az eredmény 7. bitjével megegyező érték kerül, Z értéke pedig akkor lesz 1, ha az eredmény minden bitje 0. A C bit az operandusból kiléptetett szélső bit értékét veszi fel.

Amikor az utasítás operandusa az A regiszter, a különböző assemblerek és disassemblerek eltérően működnek. Egyeseknél az operandust egy A betű jelöli, másoknál az operandus helye üresen marad. Példa az előbbire a HELP+, utóbbira a C16 beépített monitorának assemblere. Szimbolikus assemblerek használatakor célszerű előzetesen tájékozódni, hogy melyik jelölési módszert kell alkalmazni.

Alább, az utasítások ismertetésénél az A regiszterre, illetve a memóriabájtokra egységesen az operandus szóval hivatkozom.

Az ASL utasítás

Neve az angol Arithmetic Shift Left (aritmetikai eltolás balra) kifejezés kezdőbetűiből származik: az operandus minden egyes bitjét egy hellyel balra lépteti. A jobb szélső bit helyére 0 lép be, a bal szélső bit tartalma a C bitbe kerül.

Az utasítás hatása a 2-vel való szorzásnak felel meg. Több bájtra kiterjedő számításokhoz a ROL utasítással együtt kell használni. Erre rövidesen gyakorlati példát is láthatunk éppúgy, mint az ASL utasításnak az indexelt címzéssel kapcsolatos speciális alkalmazására.

LSR

Neve a Logical Shift Right (logikai eltolás jobbra) kezdőbetűiből származik. Működése megfelel az ASL utasításénak, de itt a léptetés jobbra történik, a bal szélső bit helyére lép be a 0, és a jobb szélső bit tartalma kerül C-be. Az utasítás hatása megfelel a 2-vel való osztásénak, az osztás maradéka kerül a C bitbe. Több bájtos osztáshoz a ROR-ral együtt kell használni.

ROL és ROR

Nevük a ROtate Left (forgatás balra), illetve a ROtate Right (forgatás jobbra) kifejezésekből ered. Működésük csak annyiban különbözik a hasonló eltoló utasításokétól, hogy a belépő bit értéke nem fix 0, hanem a C bitnek a végrehajtás előtti tartalma. A kiléptetett bit tartalma ezeknél az utasításoknál is a C jelzőbitbe kerül.

A gépi kódú program működése

Az előző alkalommal közölt gépi kódú rutinnak a különböző géptípusokra vonatkozó változatai csak az utasítások operandusaiban térnek el egymástól, minden egyéb megegyezik, így ez a leírás mind a három változatra egyformán érvényes. A magyarázatoknál — a korábbiakhoz hasonlóan — az utasításcímekre hivatkozom.

$033C. . .$0340: a regiszterekbe töltjük a képernyő- és a színmemória kezdőcímét. Az X regiszterbe kerül a képernyőmemória kezdőcímének magasabb helyértékű bájtja, az Y-ba a színmemóriáé. A-ba a közös alsó bájt kerül.

$0342. . .$0348: a kezdőcímeket a nullás lapon levő felhasználói területre írjuk, itt lesznek az indirekt indexelt utasítások báziscímei.

$034A: az X regiszterbe tesszük a képernyő sorainak a számát. Az X regisztert most is számlálásra használjuk. Ha tartalma 0-ra csökken, a program befejezte a feladat végrehajtását.

$034C: Y-ba az oszlopok számánál eggyel kisebb számot töltünk. A báziscímek minden ciklusléptésben az aktuális sor első pozíciójára mutatnak, az indexelés 0-ról indul. A csillagok kiírása minden sorban hátulról kezdődik.

$034E. . .$0354: a ciklusmag. Először az A regiszterbe töltjük a csillag ASCII kódját, beírjuk a képernyőmemória megfelelő pozíciójába, majd ugyanígy a fekete szín kódját is a színmemória aktuális pozíciójába írjuk.

$0356. . .$0357: a belső ciklus záró utasításpárja.

$0359. . .$0366: a báziscímek növelése. Ez részletesebb magyarázatot kíván, részint az ADC utasítás, részint pedig az itt alkalmazott programozási fogás miatt.

Betöltjük a képernyősor báziscímének alsó bájtját az A regiszterbe. A C bit törlése után hozzáadjuk a képernyőoszlopok számát, majd visszaírjuk a báziscímre. Mivel a képernyő- és a színmemória egymásnak megfelelő címeinek alsó bájtja megegyezik egymással, ugyanezt az értéket tesszük le a színmemóriasor báziscímének alsó bájtjára is. Ha az ADC utasítás végrehajtásakor átvitel is volt, INC utasításokkal megnöveljük a báziscímek felső bájtjait.

$0368. . .$036B: a külső ciklus záróutasítása, és visszatérés BASIC-be.

Miért gyorsabb, ha fekete?

A múlt alkalommal említettem, hogy ha a BASIC változatban a képernyőre írt karakterek színe fekete, a program gyorsabban fut, mit más színnel. Vajon miért? Mert a fekete szín kódja 0. Az interpreternek a valós számokat feldolgozó rutinja a 0 valós számot egyszerűbben, tehát gyorsabban kezeli, mint más valós számokat. Ha megpróbálunk valós változó helyett egész típusút használni, meglepetésünkre a program még lassúbb lesz. Ennek a többszöri konverzió az oka.

BARNA LÁSZLÓ

Közületek figyelem!

Mikroszámítógépet akarnak vásárolni? Tájékozódjanak a naprakész piaci helyzetről!

Díjtalan ismertető!

MESZ
Számítástechnika
1368. Budapest, Pf. 193.

Pontosabb számítások Pascalban

A Sinclair ZX-Spectrumra készült HISOFT Pascal kiváló fordítóprogram. Gyorsasága, könnyű kezelhetősége és szolgáltatásai méltán teszik az egyik legnépszerűbb programnyelvvé a Spectrum-felhasználók táborában. Mégis van néhány olyan sajátsága, amely bizonyos feladatok programozását akadályozza. Egyik ilyen hátránya a lebegőpontos (valós) számokban elérhető csekély pontosság.

A ROM-ba beégetett BASIC interpreter egységesen ötbájtos számokkal dolgozik, ahol négy bájt a mantissza és egy bájt a karakterisztika. Ezzel a módszerrel decimális (tizes alapú) számrendszerben kifejezve, a számok 8-9 számjegy pontosságig ábrázolhatók. A Pascal ezzel szemben a lebegőpontos számokat csak négy bájtban tárolja, melyekből három a mantissza és egy a karakterisztika, így csak 6-7 decimális számjegy pontosságú számábrázolást tesz lehetővé.

Nos, ez már igen sok esetben nem elégséges. Gondoljunk például arra, hogy ezzel a pontossággal már nem készíthetünk olyan programot, mely egy boltocska napi vagy heti forgalmát nyilvántartja. Ugyancsak gondot okoz a kis pontosság olyan numerikus módszerek alkalmazásánál, ahol a kerekítési hiba halmozódására számíthatunk. Ilyen esetekben ugyanis az értékes számjegyek száma kisebb, mint amennyit ábrázolni lehet, és ez rossz esetben oda is vezethet, hogy az eredmény teljesen hibássá torzul. E cikk végén példát is mutatunk ilyen esetre.

A probléma megoldását a hosszabb (pontosabb) számok és aritmetika használata jelenti. Ezt elvileg kétféleképpen valósíthatjuk meg: írhatunk teljesen új programrészeket gépi kódban vagy Pascalban, melyek a kívánt pontosságú számtani műveleteket hajtják végre, vagy felhasználhatjuk a BASIC interpreterben eleve rendelkezésre álló ún. lebegőpontos kalkulátor rutinjait. Most az utóbbi lehetőség megvalósítását mutatjuk be.

Feltételezzük, hogy az olvasó ismeri a Pascal nyelvet és a gépi kódú programozás elemeit, valamint a Spectrum ROM rutinjainak működését. De ezek híján se riadjon vissza: a programlista gondos bebillentyűzés után így is működni fog. Figyelmébe ajánljuk mindenesetre az alábbi könyveket:

— Dr. Bakos Tamás: Pascal PC-seknek

— Varga Imre: Pascal Spectrumra és Commodore 64-re

```
{
C122   20 PROGRAM DPCALC;
C122   30 {Kezdette Dr Kabolay Peter 1986. szeptember 11.}
C122   40 {Ez a program lineáris egyenletrendszert old meg a BASIC ROM-rutinjaiv
al}
C122   50
C122   60     CONST NN=30;
C122   70           NN1=31;
C122   80
C122   90 TYPE DREAL=ARRAY[1..5]OF CHAR;
C122  100
C122  110 VAR a:ARRAY[1..NN,1..NN]OF REAL;
C122  120     A:ARRAY[1..NN,1..NN]OF DREAL;
C122  130     i,j,N:INTEGER;
C122  140
C122  150 PROCEDURE REDRE(R:REAL;VAR D:DREAL);
C12E  160 {REAL szamot atolvas duplapontos DREAL szamba}
C12E  170 {T.Bakos "Machine Code Calculator" (ZX Computing 1986. julius/aug
ust)}
C12E  180
C12E  190 VAR B:ARRAY[1..5]OF REAL;
C12E  200     I,EXPONENT:INTEGER;
C12E  210     A,E,MANTISSA:REAL;
C12E  220     C:BOOLEAN;
C12E  230
C12E  240 PROCEDURE INDRE(N:REAL);
C131  250 BEGIN
C149  260   B[1]:=N;
C178  270   IF N<0 THEN B[2]:=255
C1C9  280   ELSE B[2]:=0;
C1D5  290   IF B[2]=255 THEN R:=R+65536;
C208  300   B[4]:=ENTIER(R/256);
C22B  310   B[3]:=R-256*B[4];
C329  320   B[5]:=0
C350  330 END;{INDRE}
C365  340
C365  350 BEGIN
C370  360   C:=FALSE;
C381  370   IF ABS(R)<MAXINT THEN
C396  380     IF R=ENTIER(R) THEN INDRE(R) ELSE C:=TRUE
C3F3  390   IF (ABS(R)>=MAXINT)OR(C=TRUE) THEN
C41A  400   BEGIN
C42A  410     EXPONENT:=1+ENTIER(LN(ABS(R))/LN(2));
C45B  420     IF EXPONENT>=-127 THEN
C479  430     BEGIN
C479  440       E:=1
C486  450       FOR I:=1 TO ABS(EXPONENT) DO
C489  460         E:=E*2;
C4E3  470 IF EXPONENT<0 THEN E:=1/E;
C510  480       MANTISSA:=ABS(R)/E;
C547  490       B[1]:=EXPONENT+128;
C583  500       A:=MANTISSA;
C59D  510       FOR I:=2 TO 5 DO
C5C9  520       BEGIN
C5C9  530         A:=A*256;
C5EA  540         B[I]:=ENTIER(A);
C625  550         A:=A-B[I];
C678  560       END;
C67D  570       C:=A>=0.5;
CA97  580       FOR I:=5 DOWNTO 2 DO
CA89  590       BEGIN
CA8C  600         IF C THEN B[I]:=B[I]+1;
CF2E  610         C:=B[I]>=256;
CF68  620         IF C THEN B[I]:=0;
CF82  630       END;
CF85  640       IF C THEN
CF8C  650       BEGIN
                  B[2]:=128;
                  B[1]:=B[1]+1
                END;
D020  670     END;
D038  680     IF R<0 THEN B[2]:=B[2]+128
D0B4  690   END ELSE
D0CA  700   BEGIN
D0CA  710     R:=R;
D0DB  720     INDRE(R)
D0EA  730   END
D0EB  740 END;
D0EB  750 FOR I:=1 TO 5 DO
D0FE  760   D[I]:=CHR(ENTIER(B[I]));
D15E  770 END;{REDRE}
D172  780
D172  790 FUNCTION dreal(D:DREAL):REAL;
D175  800 {DREAL duplapontos szamot atalvas REAL szamba}
D175  810 {T.Bakos "Machine Code Calculator" (ZX Computing 1986. julius/aug
ust)}
D175  820 VAR I,SIGN,EXPONENT:INTEGER;
D175  830     E,MANTISSA:REAL;
D175  840     C:ARRAY[1..5]OF INTEGER;
D175  850 BEGIN
D18D  860   FOR I:=1 TO 5 DO
D198  870     C[I]:=ORD(D[I]);
DA88  880   IF C[1]=0 THEN
DA51  890   BEGIN
DA51  900     E:=C[5]+C[4]*C[3];
DA97  910     IF C[2]=255 THEN
DAC8  920       E:=E-65536;
DADE  930   END ELSE
DAF2  940   BEGIN
DAF2  950     EXPONENT:=C[1]-128;
DB23  960     SIGN:=1;
DB3F  970     IF C[2]>128 THEN
DB42  980     BEGIN
DB52  990       SIGN:=-1;
DB6F 1000       C[2]:=C[2]-128;
DB9A 1010     END;
DB9A 1020     MANTISSA:=1;
DBC2 1030     FOR I:=2 TO 4 DO MANTISSA:=MANTISSA+C[I]/256;
DBCE 1040     MANTISSA:=(MANTISSA+C[1])/256;
DC44 1050     E:=1;
DC56 1060     FOR I:=1 TO ABS(EXPONENT) DO
DC83 1070       E:=E*2;
DCAD 1080     IF EXPONENT<0 THEN E:=1/E;
DCE7 1090     D:=SIGN*E*MANTISSA
DD03 1100   END;
DD1E 1110   dreal:=D
DD1E 1120 END;{dreal}
DD45 1130
DD45 1140 PROCEDURE DREALToStack(A:DREAL);
DD4B 1150 {DREAL szamot a kalkulator stackjbe tolt}
DD4B 1160 BEGIN
DD4B 1170   INLINE(/DD,/7E,/0,  {LD A,(IX+0)
DD63 1180              /DD,/56,/3,  {LD E,(IX+3)
DD6A 1190              /DD,/56,/4,  {LD D,(IX+4)
DD71 1200              /DD,/46,/5,  {LD C,(IX+5)
DD80 1210              /DD,/66,/6,  {LD B,(IX+6)
DD8F 1220              /CD,/8A,/2A {CALL /2A8A})
DD9E 1230 END;{DREALToStack}
DD9E 1240
DD9E 1250 PROCEDURE StackToDREAL(VAR A:DREAL);B:DREAL;
DD9E 1260 {A stackbol egy DREAL szamot kiolvas}
DD9E 1270 BEGIN
DD9E 1280   INLINE(/CD,/F1,/28,{CALL /28F1
DDA4 1290              /DD,/77,/3,  {LD (IX+2),A}
```

```
CD9D 1300            ADD,F3,3,   (LD (IX+3),E)
CDA0 1310            ADD,F2,4,   (LD (IX+4),D)
CDA3 1320            ADD,F1,5,   (LD (IX+5),C)
CDA6 1330            ADD,F0,6    (LD (IX+6),B));
CDA9 1340    A:=0
CDAB 1350 END;(StackToDREAL)
CDC7 1360
CDC7 1370 PROCEDURE DADD(B,C:DREAL;VAR D:DREAL);
CDCA 1380 (DREAL szamok osszeadasa:D=B+C)
CDCA 1390 BEGIN
CDE2 1400   DREALToStack(B);
CDE8 1410   DREALToStack(C);
CDEE 1420   INLINE(!EF,          (RST /28)
CDEF 1430            /BF,        (Osszeadas)
CDF0 1440            /38         (Szamitas vege));
CDF1 1450   StackToDREAL(D,B)
CDF5 1460 END;(DADD)
CDF8 1470
CDF8 1480 PROCEDURE DSUB(B,C:DREAL;VAR D:DREAL);
CDFB 1490 (DREAL szamok kivonasa:D=B-C)
CDFB 1500 BEGIN
CE0A 1510   DREALToStack(B);
CE0F 1520   DREALToStack(C);
CE17 1530   INLINE(!EF,          (RST /28)
CE18 1540            3,          (Kivonas)
CE19 1550            /38         (Szamitas vege));
CE1A 1560   StackToDREAL(D,B)
CE1A 1570 END;(DSUB)
CE1D 1580
CE1D 1590 PROCEDURE DMULT(B,C:DREAL;VAR D:DREAL);
CE2C 1600 (DREAL szamok szorzasa:D=B*C)
CE2C 1610 BEGIN
CE34 1620   DREALToStack(B);
CE12 1630   DREALToStack(C);
CF3B 1640   INLINE(!EF,          (RST /28)
CF3A 1650            4,          (Szorzas)
CF32 1660            /38         (Szamitas vege));
CF33 1670   StackToDREAL(D,B)
CF4F 1680 END;(DMULT)
CF62 1690
CF62 1700 PROCEDURE DDIV(B,C:DREAL;VAR D:DREAL);
CF65 1710 (DREAL szamok osztasa:D=B/C)
CF65 1720 BEGIN
CF7D 1730   DREALToStack(B);
CF9B 1740   DREALToStack(C);
CF89 1750   INLINE(!EF,          (RST /28)
CF8A 1760            5,          (Osztas)
CF8B 1770            /38         (Szamitas vege));
CF8C 1780   StackToDREAL(D,B)
CF8B 1790 END;(DDIV)
CFEB 1800
CFEB 1810 PROCEDURE GAUSS(N:INTEGER);
CFEE 1820 (Lineris egyenletrendszer megoldasa Gauss-eliminacioval)
CFEE 1830 (Lasd Gyulai "Az ALGOL AB programozasi nyelv" MK 1971. c. konyve alap
on)
CFEE 1840 VAR I,J,K:INTEGER;
CFEE 1850     T,M:REAL;
CFEE 1860 BEGIN
D086 1870   FOR I:=1 TO N DO
D030 1880     BEGIN
D033 1890       M:=1/A(I,I);
D08F 1900       FOR K:=I+1 TO N+1 DO
D0DE 1910       A(I,K):=M*A(I,K);
D163 1920       FOR J:=1 TO N DO
D18D 1930       IF J<>I THEN
D1AB 1940         BEGIN
D1AB 1950           T:=A(J,I);
01F9 1960           FOR K:=I+1 TO N+1 DO
0228 1970             A(J,K):=A(J,K)-T*A(I,K)
02FC 1980         END
D318 1990     END
D31F 2000 END;(GAUSS)
D32F 2010
```

1. lista

```
D32F 2020 PROCEDURE DGAUSS(N:INTEGER);
D332 2030 (Lineris egyenletrendszer megoldasa Gauss-eliminacioval 5 byteos arit
metikaval)
D332 2040 VAR I,J,K:INTEGER;
D332 2050     B:REAL;
D332 2060     T,M,S,Z:DREAL;
D332 2070 BEGIN
D34A 2080   Z:=1.0;
D35C 2090   REDRE(0,B);
D37B 2100   FOR I:=1 TO N DO
D3A5 2110   BEGIN
D3A8 2120     DDIV(B,A(I,I),M);
D41C 2130     FOR K:=I+1 TO N+1 DO
D44B 2140     DMULT(M,A(I,K),A(I,K));
D4FF 2150     FOR J:=1 TO N DO IF J<>I THEN
D544 2160     BEGIN
D544 2170       T:=A(J,I);
D572 2180       FOR K:=I+1 TO N+1 DO
D5C1 2190       BEGIN
D6C4 2200         DMULT(T,A(I,K),Z);
D638 2210         DSUB(A(J,K),Z,A(J,K))
D6DC 2220       END
D6E5 2230     END
D6E9 2240   END
D6ED 2250 END;(DGAUSS)
D6FD 2260
D6FD 2270 BEGIN
D70A 2280   PAGE;
D70D 2290   WRITE('Az egyenletek szama: ');
D728 2300   READ(N);
D731 2310   WRITELN;
D734 2320   FOR I:=1 TO N DO
D752 2330   BEGIN
D755 2340     WRITELN;
D758 2350     WRITELN(I,'. egyutthato sor');
D77F 2360     WRITELN;
D782 2370     FOR J:=1 TO N DO
D7A8 2380     BEGIN
D7A3 2390       WRITE('  ',J,'.      ');
D7CB 2400       READ(A[I,J]);
D818 2410       REDRE(A[I,J],A[I,J]);
D8A1 2420     END;
D8B5 2430     WRITELN;
D8B8 2440     WRITE('Jobboldali: ');
D8AE 2450     READ(A[I,N+1]);
D8F4 2460     REDRE(A[I,N+1],A[I,N+1]);
D977 2470   END;
D979 2480   PAGE;
D988 2490 WRITE(CHR(22),CHR(8),CHR(8));
D995 2500   WRITELN('EREDMENYEK:');
D9AE 2510   WRITELN('(BASIC othajtos pontossag)');
D9D6 2520   WRITELN;
D9D9 2530   DGAUSS(N);
D9E2 2540   FOR I:=1 TO N DO
DA08 2550   WRITELN(I:2,'. ',drere(A[I,N+1]));
DA79 2560   WHILE INKEY$<>'' DO;
DA8F 2570   GAUSS(N);
DA98 2580   WRITELN;
DA98 2590   WRITELN('EREDMENYEK:');
DAB4 2600   WRITELN('(PASCAL negybajtos pontossag)');
DADF 2610   WRITELN;
DAE2 2620   FOR I:=1 TO N DO
DB08 2630   WRITELN(I:2,'. ',A[I,N+1])
DB08 2640 END.
End Address: DB69
Run?C
```

2. lista

```
  10 INPUT "n=";n
  20 DIM a(n,n+1)
  30 PRINT "Egyutthatok:""
  40 FOR i=1 TO n
  50 FOR j=1 TO n
  60 PRINT i;". sor, ";j;". elem: ";
  70 INPUT a(i,j)
  80 PRINT a(i,j)
  90 NEXT j
 100 PRINT
 110 PRINT "i:".sor jobboldal: ";
 120 INPUT a(i,n+1)
 130 PRINT a(i,n+1)
 140 PRINT
 150 NEXT i
1000 FOR i=1  TO n
1010 LET m=1/a(i,i)
1020 FOR k=i+1 TO n+1
1030 LET a(i,k)=m*a(i,k)
1040 NEXT k
1050 FOR j=1 TO n
1060 IF j=i THEN  GO TO 1100
1070 LET T=a(j,i)
1080 FOR k=i+1 TO n+1
1090 LET a(j,k)=a(j,k)-T*a(i,k)
1100 NEXT k
1110 NEXT j
1120 NEXT i
2000 CLS : PRINT "EREDMENYEK:"
2010 FOR i=1 TO n
2020 PRINT i;TAB 4;a(i,n+1)
2030 NEXT i
```

— ZX-Spectrum ROM programja.

Vizsgáljuk meg, milyen részfeladatokat kell megoldanunk:

— A Pascal program elején öt karakterből álló DREAL típust definiálunk. Ilyen DREAL típusú változókban fogjuk tárolni az ötbájtos, BASIC típusú lebegőpontos számokat.

— Egy-egy típuskonverziós eljárást illetve függvényt írunk, melyek a Pascal négybájtos REAL számait számítják át DREAL típusúba és viszont.

— Egy-egy eljárást szerkesztünk, melyek közvetlenül az egyes aritmetikai műveletek előtt a DREAL számokat betöltik az ún. lebegőpontos kalkulátor veremtárába (stack), illetve a veremtárból visszatöltik a DREAL változókba.

— Egy-egy eljárást írunk a veremtár tetején elhelyezkedő két ötbájtos szám összeadására, kivonására, összeszorzására, elosztására.

Az *1. listán* egy Pascal forrásnyelvű program látható. Magáról a programról még beszélünk, most csak az egyes eljárásokra és függvényekre forditsuk figyelmünket.

A REDRE eljárás REAL számot alakít át DREAL típusúvá. A rutin megértéséhez előbb tisztáznunk kell, hogy a BASIC milyen formában tárolja a számokat. Az interpreter három esetet különböztet meg. A kis pozitív egész számokat (vagyis azokat, amelyek 0 és 65 535 közé esnek) a BASIC „0,0,k,n,0" alakban ábrázolja, ahol „n" a nagyobb helyiértékű bájtot, „k" pedig a kisebb helyiértékű bájtot jelenti. Például 258-at a BASIC így tárolja: „0,0,2,1,0", mivel 258 = 1*256 + 2.

A −65 535 és −1 közé eső kis egész számok alakja hasonló, csak a második bájtba

0 helyett 255-öt kell írnunk, tehát a forma ilyen: „0,255,k,n,0". A 0 szám alakja: „0,0,0,0,0".

Minden más lebegőpontos vagy egész számot a következő szabályok szerint tárol a BASIC. Az első bájt a karakterisztika +128, a következő négy bájt pedig a mantissza egymás után következő bájtjait tartalmazza. A mantissza abszolút értéke mindig 0,5 és 1 közé esik. Ha a szám negatív, a második bájt 7. bitje 1, ha pozitív, akkor 0. Ez más szóval azt jelenti, hogy a negatív szám második bájtjához 128-at hozzá kell adnunk.

A fentiek ismeretében a REDRE eljárás már érthető lesz. A 370-es sor megvizsgálja, hogy a kérdéses R szám beleesik-e a BASIC kis integer számtartományába. Ha igen, a program az INDRE eljárással végrehajtja az ötbájtos konverziót. Ha nem, megvizsgálja, hogy a szám abszolút értéke kisebb-e 2 exp—127-nél (ez a BASIC-ben ábrázolható legkisebb abszolút értékű szám); ha igen, az ötbájtos számot nullákkal tölti fel (410-es és 690—730-as sorok).

Minden más esetben az eljárás a fent leírt szabályoknak megfelelően kiszámítja előbb a karakterisztikát, majd a mantissza 4 bájtját.

A drere függvény a REDRE inverz műveletét valósítja meg, vagyis ötbájtos számot a Pascalban használt négybájtossá alakít. Itt a program előbb megvizsgálja, hogy a kérdéses szám kis integer-e (ha a DREAL szám első bájtja nulla, akkor igen), és a vizsgálat eredményétől függően a 890—930-as, illetve lebegőpontos szám esetén a 940-1100-as sorokban látható algoritmust hajtja végre.

A DREALToStack eljárás, mint a neve is mutatja, a BASIC lebegőpontos kalkulátor-veremtárába tölti az A DREAL típusú számot. Az eljárás rövid gépi kódú szubrutint tartalmaz INLINE függvénybe ágyazva. Tudni kell, hogy a HISOFT Pascalban függvény vagy eljárás hívásakor az érték szerint hívott paraméter bájtjai a Z80 mikroprocesszor IX regisztere által címzett 2. bájttól felfelé helyezkednek el. Tehát esetünkben az A formális DREAL paraméter az IX+2—IX+6 címeken található.

Az ötbájtos számoknak a veremtárba való töltésére szolgáló gépi kódú szubrutint nem kell megírnunk, mert ilyet a ROM tartalmaz. A szubrutin címe hexadecimális jelöléssel 2AB6, hívása előtt a Z80 A, E, D, C, B reigsztereit az adott sorrendben fel kell tölteni az ötbájtos számmal. Nos, az INLINE függvény az 1170—1230-as sorok között éppen ezt teszi.

A StackToDREAL eljárás, mely a veremtárból az A DREAL változóba tölt egy hosszú lebegőpontos számot, hasonlóan működik. A gépi kódú szubrutin címe 2BF1, és a visszatéréskor az előbbi regiszterek tartalmazzák a szám egyes bájtjait. A paraméterátadás problémáját egy kis trükkel oldottuk meg: a formális paraméterek listájára felvettünk egy látszólag felesleges B érték szerint hívott DREAL változót, melybe híváskor bármit beírhatunk. Erre csak azért van szükség, hogy a Z80 regisztereiben tárolt szám bájtjait ideiglenesen tárolja. A főprogramnak a számot a név szerint hívott DREAL A paraméter adja át.

Az elemi aritmetikai műveletek a DADD, DSUB, DMULT és DDIV eljárások segítségével végezhetők el. Mindegyik hasonló felépítésű: előbb betöltik az A és B érték szerint hívott DREAL paramétert a lebegőpontos kalkulátor veremtárába, majd az RST 28 (hexadecimális cím!) segítségével hívjuk a megfelelő ROM-rutint. Ezután az eredményt, mely a veremtár tetején található, a D név szerint hívott paraméternek adja át a program. Figyeljük meg, hogy szemben a USER függvénnyel, az INLINE függvény gépi kódú szubrutinját nem kell RET utasítással lezárni.

Végül lássunk egy gyakorlati példát az ismertetett eljárások használatára. A mintaprogramban többismeretlenes lineáris egyenletrendszer megoldását mutatjuk be az ún. Gauss-eliminációs módszerrel.

A program kétféleképpen is megoldja az egyenletrendszert: a négybájtos Pascal valós számok felhasználásával (GAUSS eljárás), valamint a BASIC ötbájtos számaival (DGAUSS eljárás). Összehasonlítás céljából elkészítettük a Gauss-elimináció BASIC programját is *(2. lista)*. Próbafuttatások alatt egy 20 ismeretlenes egyenletrendszer megoldásához a BASIC programnak 125 másodpercre, a BASIC ötbájtos számait használó Pascal rutinnak 22 másodpercre, a négybájtos Pascal számokkal dolgozó eljárásnak 6 másodpercre volt szüksége. A két első program eredményei teljesen megegyeznek, a Pascal eredménye viszont olyan pontatlan, hogy gyakorlatilag használhatatlan. Ennek az állításnak az igazáról az eredmények visszahelyettesítésével meggyőződhetünk.

Úgy gondoljuk, a fenti tények mindenkit meggyőznek a módszer hasznosságáról.

A ROM lebegőpontos kalkulátor rutinjai igen sok további lehetőséget kínálnak a Pascalon belüli felhasználásra.

DR. KABOLDY PÉTER

Z80 programok haladóknak Spectrumra és Primóra

2. Manókezelő program

Az *1. lista* programja segítségével tetszés szerinti számú sprite (manó) rakható ki a képernyőre úgy, hogy az egyes manók a megfelelő sorrendben takarják egymást is és a hátteret is. Ez azt jelenti, hogy a találkozó manók nem keverednek csúnyán összse, hanem a fölül levő manó teljesen letakarja az alatta levőt, ahol nem átlátszó. Külön erénye a programnak, hogy mindezt villogásmentesen teszi meg. Színek kezelésére sajnos nincs lehetőség, viszont a teljes fedés miatt különböző mintázatokkal jól lehet pótolni őket.

A program 32*32 pixel méretű manókat kezel. A manók alakját a következőképpen kell megadni.

A manó által letakart felület alakját letessszük egy x címtől, balról jobbra sorfolytonosan. Egy sor éppen négy bájt lesz. Fontos, hogy x lap elejére mutasson. (x legyen osztható 256-tal.) Ez a maszk 128 bájtot foglal. Ebben az egyes bit jelent átlátszót, a nullás pedig elfedőt!

A maszk után közvetlenül, vagyis x+128 címre kell tenni a manó képét, hasonlóan a maszkhoz. Ez is 128 bájt.

Tehát egy manó kinézete éppen 256 bájtot, egy lapot foglal. Akárhány ilyen manóábránk lehet. Ez logikailag nem azonos a kirajzolt manóval: több manónak lehet ugyanaz a képe, de lehet olyan manónk is, amelyhez több alak tartozik, például a mozgó manó mozgásfázisai.

Létrehozunk a memóriában egy olyan adatmezőt, amely az egyes kirajzolandó manók adatát tartalmazza („manófájl"). Ennek első bájtja az egyes elemek hosszát mutatja. Utána következnek az egyes manók adatai, ezek a manóelemek. Az utolsó manóelem után a fájlvégjel egy db 0-ás bájt. A manóelemek szerkezete:

a) a manó alakjának címe — 1 bájt (lapsorszám);

b) x koordináta (0 . . . 223) 1 bájt;

c) y koordináta (−1 . . . 160) 1 bájt;

Ha y=−1, akkor az a manó nem látható a képen.

d) a fennmaradó bájtokon saját adatokat tárolhatunk. Ha erre nincs szükség, akkor az elemhossz 3 legyen.

Példa a manófájlra: #03, #F0, #00, #00, #F1, #64, #32, #00. (Itt a '#' jel a hexadecimális számokat jelöli.) Jelentése: a manóelem-hossz 3 bájt, két manót kell kirakni, éspedig az első alakja #F000 címen található, és x=0, y=0 helyre kell kirajzolni. A második alakja a #F100 cí-

```
10 *C-                        ; GENS-nek szóló: ne listázzon.
60         ORG  $C000         ; például
65 SCREEN  EQU  $4000
70 PUFC    DEFS $1800         ;cime $800-zal osztható legyen
75 PUFFER  EQU  $80           ; (PUFC-SCREEN) felső bájtja !
80 HATTER  DEFS $1800         ;cime tetszőleges lehet
110 ;    $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
120 ;    $                                             $
130 ;    $    Specsprite - 5a   32*32-es manókezelő    $
140 ;    $              UHI-software BME               $
150 ;    $                 1987.03.01                  $
160 ;    $                                             $
170 ;    $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
180 LDIRG  LDI                ;gyors LDIR
190        PUSH BC            ;ide kerül 32 db LDI,
200        PUSH DE
210        PUSH HL
220        LD   BC,62
230        LD   DE,LDIRG+2
240        LD   HL,LDIRG
250        JP   CREATE
260        DEFS 47            ;egészen idáig
270        LD   A,B
280        OR   A
290        JP   NZ,LDIRG
300        LD   A,C
310        CP   32
320        JP   NC,LDIRG
330        RET
340 CREATE LDIR               ;önmódosító rész
350        POP  HL
360        POP  DE
370        POP  BC
380        JR   LDIRG+2       ;az első LDIt végrehajtotta
390 ;---------------------------------------------------
400 ;  B=y  C=x  IX=manóalak cime
410 ;  A pufferképre kitesz egy manót.
420 ;---------------------------------------------------
421 SPRITE LD   A,B           ;ha y koordináta -1, akkor azt
422        INC  A             ; a manót nem kell megjeleni-
423        RET  Z             ; teni visszatér
430        PUSH DE            ;minden használt regisztert
440        PUSH HL            ; elment
450        EXX
460        PUSH DE
470        PUSH HL
480        EXX
490        LD   DE,$0080      ;IX-et az alakra állítja rá
500        ADD  IX,DE
510        LD   A,$9F         ;a ROMbeli pix_add rutint
520        CALL PIXADD        ; 159-es y korláttal hivja
530        LD   C,A           ;szükséges jobbra tolások
540        LD   A,PUFFER      ;a kép és a puffer távol-
550        ADD  A,H           ; sága lapokban.
560        LD   H,A
570        PUSH IY
580        PUSH HL
590        POP  IY            ;a pufferbeli cim IY-ba kerül
600        LD   B,32          ;a manó 32 sorból áll
610 LOOP0  PUSH BC            ;a manósorokat rajzoló ciklus
620        LD   A,C           ;szükséges tolások száma
630        LD   C,(IX-128)    ;a manóalak B,D,H,D',H'-be
640        LD   B,(IX+0)      ; a maszk C,E,L,E',L'-be kerül
650        LD   E,(IX-127)
660        LD   D,(IX+1)
670        LD   L,(IX-126)
680        LD   H,(IX+2)
690        EXX
700        LD   E,(IX-125)
710        LD   D,(IX+3)
720        LD   HL,$00FF      ;tolás előtt a maszk és kép
730        EXX                ; jobb oldala
740        OR   A             ;tolások
750        JR   Z,E1          ;ha nem kell tolni
760 C1     SRL  B             ; $
770        RR   D             ; $
780        RR   H             ; $ ez a tolóciklus:
790        EXX                ; $ a sprite egy egész sorát
800        RR   D             ; $ azaz 5 bájtot egyszerre tol.
810        RR   H             ; $
820        EXX                ; $
830        SCF                ;maszkba "átlátszó" jön balról
840        RR   C             ; *
850        RR   E             ; *
860        RR   L             ; *
870        EXX                ; * ez egy sornyi maszk tolása.
880        RR   E             ; *
890        RR   L             ; *
900        EXX                ; *
910        DEC  A
920        JP   NZ,C1         ;tolóciklus vége
930 E1     LD   A,(IY+0)      ;képernyő eredeti tartalma
940        AND  C             ;maszk
950        OR   B             ;sprite-adat
960        LD   (IY+0),A      ;képernyő új tartalma.
970        LD   A,(IY+1)      ;ezt egy sprite-sorra meg-
980        AND  E             ; csinálja - ez 5 bájt.
990        OR   D
1000       LD   (IY+1),A
1010       LD   A,(IY+2)
1020       AND  L
1030       OR   H
1040       LD   (IY+2),A
1050       LD   A,(IY+3)
1060       EXX
1070       AND  E
1080       OR   D
1090       LD   (IY+3),A
1100       LD   A,(IY+4)
1110       AND  L
1120       OR   H
1130       LD   (IY+4),A
1140       EXX
1150       LD   BC,4          ;manóalak következő sora
1160       ADD  IX,BC
1170       CALL IYDOWN        ;lefelé lépés a pufferképen
1180       POP  BC
1190       DEC  B             ;sorok-ciklus vége
1200       JP   NZ,LOOP0
1210       POP  IY            ;visszahozza az elmentett
1220       EXX                ; regisztereket
1230       POP  HL
1240       POP  DE
1250       EXX
1260       POP  HL
1270       POP  DE
1280       RET
1290 ;--------------------------------------------------
1300 IYDOWN DEFB $FD          ; IY-t lefelé viszi egy sorral
1310       INC  H             ;INC YH
1320       DEFB $FD
1330       LD   A,H           ;LD A,YH
1340       AND  7
1350       RET  NZ
1360       LD   BC,-$07E0
1370       ADD  IY,BC
1380       DEFB $FD
1390       LD   A,H           ;LD A,YH
1400       AND  7
1410       RET  Z
1420       LD   A,7
1430       DEFB $FD
1440       ADD  A,H           ;ADD A,YH
1450       DEFB $FD
1460       LD   H,A           ;LD YH,A
1470       RET
1475 ;--------------------------------------------------
1480 PIXADD EQU  $22AC        ;kiszámítja egy képelem cimét.
1485 ;--------------------------------------------------
1490 ALAPOZ PUSH HL           ;képernyőtartalmat a háttér-
1500       PUSH DE            ;tárolóba másolja
1510       PUSH BC
1520       LD   HL,SCREEN
1530       LD   DE,HATTER
1540 MOZG  LD   BC,$1800      ;képhossz
1550       CALL LDIRG         ;gyors LDIR
1560       POP  BC
1570       POP  DE
1580       POP  HL
1590       RET
1600 ;--------------------------------------------------
1610 ALAP  PUSH HL            ;hátteret a pufferbe másolja
1620       PUSH DE
1630       PUSH BC
1640       LD   HL,HATTER
1650       LD   DE,PUFC
1660       JR   MOZG          ;$1800 bájt mozgatása
1670 ;--------------------------------------------------
1680 MASOL PUSH HL            ;a puffer tartalmát a képer-
1690       PUSH DE            ; nyőre másolja
1700       PUSH BC
1710       LD   HL,PUFC
1720       LD   DE,SCREEN
1730       JR   MOZG
1740 ;--------------------------------------------------
1750 SPS   PUSH IX            ;a manófájlt olvassa és kitesz
1760       PUSH BC            ; sorban minden manót a puf-
1770       PUSH DE            ; ferre
1780       LD   D,0
1790       LD   E,(HL)        ;manófájl-elemhossz
1800       INC  HL
1810 SPS_1 LD   A,(HL)        ;manóalak cim (lapsorszám)
1820       OR   A
1830       JR   Z,SPS_E       ;ha nulla, vége a manófájlnak
1840       DEFB $DD           ;IX-be tölti a manó alakcimét
1850       LD   H,A           ;LD XH,A
1860       DEFB $DD
1870       LD   L,0           ;LD XL,0
1880       PUSH HL
1890       INC  HL
1900       LD   C,(HL)        ;BC-be a koordináták
1910       INC  HL
1920       LD   B,(HL)
1930       CALL SPRITE        ;egy manó kivitele
1940       POP  HL
1950       ADD  HL,DE         ;manófájl következő elemére
1960       JR   SPS_1
1970 SPS_E POP  DE            ;(végjel esetén ide kerül)
1980       POP  BC
1990       POP  IX
2000       RET
2010 ;--------------------------------------------------
2020 SPECSP CALL ALAP         ;háttér => puffer
2030       CALL SPS           ;manók  => pufferre
2040       JP   MASOL         ;puffer => képernyő
2050 ;--------------------------------------------------
```

1. lista

men van, és az x = 100, y = 50 helyre kell kitenni.

A manók mögé kerülő hátteret a háttérpufferbe tegyük. Ez # 1800 bájtot jelent. Ehhez segítséget nyújt az ALAPOZ szubrutin, amely a képernyőtartalmat a háttérpufferbe másolja.

A kirajzoló program meghívásakor HL a manófájl címét tartalmazza. Indítási cím: SPECSP.

```
Módositások Primohoz:

SCREEN EQU  $E800       ; az aktuális képernyőre mutasson.
HATTER EQU  ...         ;egy $1800 bájtos pufferre mutasson
PUFC   EQU  ...         ;egy $1800 bájtos pufferre mutasson
PUFFER EQU  ...         ; (PUFC-SCREEN) felső bájtja.
;-------------------------------------------------------
;   PIXADD  kiszámítja egy képelem címét a képernyőn.
;       B=y  C=x  A=y határ   SCRREN=képernyőcím
;       HL=cím a képernyőn  A=biteltolás
;-------------------------------------------------------
PIXADD SUB  B            ;túllépi-e a határt
       JP   C,ERROR      ;Izlés szerinti hibarutin
       LD   B,A          ;y-t alulról számolja.
       XOR  A            ;x alsó három bitje a bájtok elto-
       SRL  B            ; lását adja meg, becsorog az akku
       RR   C            ; ba. BC többi bitje a cím alsó 13
       RRA               ; bitjét határozza meg.
       SRL  C
       RR   C
       RRA
       SRL  C
       RR   C
       RRA
       RLCA              ; most az akkumulátor fölső 3 bit-
       RLCA              ; jét az alsó 3 helyére forgatjuk.
       RLCA
       LD   HL,SCREEN    ; a képernyő kezdőcíme.
       ADD  HL,BC
       RET
;-------------------------------------------------------
IYDOWN LD   BC,32        ;IY-t lefelé lépteti a képernyőn
       ADD  IY,BC        ; egy sorral. A sor hossza 32 bájt.
       RET
;-------------------------------------------------------
```

2. lista

Minden alkalommal, amikor (egy vagy több) változtatás történt a háttéren vagy a manófájlban, futtassuk a kirajzoló programot. A program lefutása után mindig a tárolt háttérnek és a manófájlnak megfelelő kép lesz a képernyőn.

Hátránya a programnak, hogy külön kell tárolni a hátteret, sőt, pufferképre van szükség a kép összeszerkesztéséhez. Ez nagy memóriaigénnyel jár. Ezenkívül meglehetősen lassú is: minden alkalommal újra szerkeszti a képet. Ezért ezt a programot nem célszerű kevés manó ábrázolására használni. Sok manó esetén viszont, ha egy időegység alatt több manó elmozdul vagy alakot változtat, sokkal gazdaságosabb a memóriafelhasználás, és az egy manóra jutó gépidő is rövidebb.

A program Spectrumra készült, de a *2. listán* megadott változtatásokkal Primón is fut. Mivel a programot én csak Spectrumon próbáltam ki, kérem a Primo-tulajdonosokat, hogy számoljanak be tapasztalataikról.

A forrásszöveg egyébként GENS3 assemblerhez készült. Akinek nem ilyen van, ne feledkezzen meg a szükséges átalakításokról. Ha az olvasónak nincs megfelelő assemblere, vagy a GENS-t szeretné megkapni, postafordultával visszaküldöm a hozzám eljuttatott kazettáját.

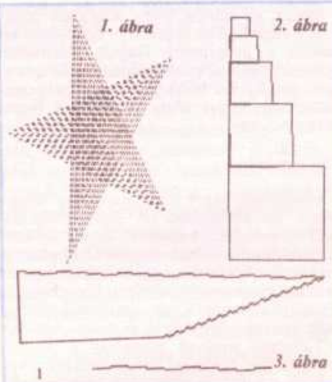
A program tartalmazza a gyors LDIR rövid változatát (LDIRQ). A lényeget a SPRITE címkéjű rutin tartalmazza. Egy IX című manót a pufferképre tesz BC koordinátákra. Az IYDOWN rutin megadja a képernyőn az IY című bájt alatt elhelyezkedő bájt címét. Ez a pufferképen is működik, mert annak a sorszerkezete a képernyőszerkezethez hasonló. Ez a rutin a Primóra írt változatban sokkal egyszerűbb, hiszen ott a kép sorai sorban egymás után következnek. Nem szorul magyarázatra az ALAP és a MÁSOL rutinok működése. Az SPS rutin az, amelyik végigolvassa a manófájlt, és aszerint meghívogatja a SPRITE rutint.

Vigyázzunk arra, hogy Spectrumon a pufferkép kezdőcíme legyen osztható #800-zal, egyébként nem működik a program. Erre azért van szükség, mert a pufferkép a képernyőhöz hasonló módon tartalmazza a pixelsorokat.

UHERKOVICH PÉTER

Teknősbéka-grafika IV.

Az előző részt azzal fejeztem be, hogy a továbblépéshez teljesen át kell alakítanunk az eddigi program szerkezetét, vagyis szinte új programot kell írnunk. Mielőtt egy ilyen nagy vállalkozásba kezdenénk, vizsgáljunk meg egy mindenféle rajzolással együttjáró problémát.



1. ábra *2. ábra* *3. ábra*

Nézzük meg az *1. lista* szerinti LOGO eljárást és a futtatás eredményeként adódó *1. ábrát.* A csillag felületes megtekintésre egész jónak tűnik, alaposabban megvizsgálva viszont az élei hepehupások. A *2. lista* szerinti eljárás és *2. ábra* szerinti képe még futólagos átnézésben is „furcsa pár", mert az ábrán a függőleges egyes esetekben megtörik, a vízszintes vonalak pedig néhol vastagítottak.

A jelenség jobban tanulmányozható és könnyebben megérthető a *3. lista* és a hozzá tartozó *3. ábra* alapján.

A program először felrajzol két szakaszt. A két egyenes szakasz egymáshoz képesti eltolását figyelmen kívül hagyva, az egyik a másikhoz százszorozott iránytangenssel és századszoros értelmezési tartománnyal állítható elő. Megnézve a két egyeneshez tartozó értékpárokat, a következők adódnak:

```
TO SZOEGSPI :OLDAL :SZOEG
FORWARD :OLDAL
RIGHT :SZOEG
SZOEGSPI (:OLDAL + 3) :SZOEG
END
```

1. lista

2. lista

```
TO TORONY :OLDAL
IF :OLDAL < 3[STOP]
NEEGYZET :OLDAL
FORWARD :OLDAL
TORONY :OLDAL * 2 / 3
END
```

```
20 FOR[illegible]=10+[illegible]-50)+150:PS
ET[illegible]:NE[illegible]=50TO180:Y=10+(X-80)
/100+[illegible]:PSET[illegible]:NEXT[illegible]
30 LINE[illegible]120).PSET:LINE-(120,1
20).PSET:LINE-([illegible]).PSET:LINE-(40,40)
,PSET
```

3. lista

1. egyenes (az eltolást levonva mindkét egyenesnél):

X Ø .01 .02 .03 .04 .05 .06 .07 .08 .09
Y Ø .1 .2 .3 .4 .5 .6 .7 .8. 9

2. egyenes

X Ø .1 2 3 4 5 6 7 8 9
Y Ø .1 .2 .3 .4 .5 .6 .7 .8 .9

Az első egyenesnek százszoros meredekségűnek kellene lenni; az ábrára rátekintve viszont teljesen mást látunk. Az első egyenes függőleges, a második pedig egy fel-le lépkedő, lefelé tartó valami.

A program második fele egy derékszögű négyszög rajzolására szolgál. A függőleges és vízszintes vonalak itt valóban függőlegesek és vízszintesek. Ez tehát rendben van. A ferde szakaszok helyett azonban itt is fel-le lépkedő, de összességében lefelé, illetve felfelé tartó rajzolatot kapunk.

Nézzük meg messzebbről az ábrát. Minél távolabbról nézzük, annál jobban „lekopnak" a lépcsők. Ha ezeket képernyőn látjuk, csak közel hajolva vesszük észre, hogy a vonal igazából nem egyenes. Mindezeket a tapasztalatainkat visszavezethetjük az olcsóbb számítógépek képernyőkezelési módjára és a képernyőre rajzoló BASIC szubrutinok sajátságaira. Ezeknél az eszközöknél a képernyőre a gép viszonylag kis sűrűségű pontok formájában rajzol (és ír, mert a szövegek karaktereit is rajzolja, csak a rajz itt előre el van készítve). A pontok a képernyőt nézve messziről egybeolvadnak, viszont közelről megkülönböztethetők. A rajzoknál mindkét tengely irányában nyolcszoros nagyítást használtam, így minden jobban láthatóvá vált.

A rajzoló rutinok kiszámítják a pontok helyét, és elhelyezik a képernyőn. Minél nagyobb a számítás pontossága, annál bonyolultabb és lassúbb a rutin, és a rajzolás sebessége, pontossága gépenként, szoftverenként eltérő. Ugyanez vonatkozik az egyeneseket (esetleg más alakzatokat) rajzoló rutinokra is. Általában más a megoldása a vízszintes és függőleges egyenesek rajzolásának, mint a ferdékének. Így lehetséges az, hogy a 3. ábra vízszintes és függőleges egyenesei jók, míg a két másik egyenes nem. (Ezzel a problémával a Microsoft BASIC rajzoló rutinjait tárgyaló cikkünkben is foglalkozni fogunk.)

A 2. ábra függőleges és vízszintes egyenesei jól szemléltetik, hogy ha a rajzoló rutin nem „tudja", hogy függőlegest, illetve vízszintes rajzol, akkor a vízszintes és függőleges is pontatlan.

DR. SIMONYI ENDRE

Toronyórát lánccal...

A programozási nyelvek szaporodása és elterjedése — úgy tűnik — mindenki számára lehetővé teszi, hogy közülük ízlésének megfelelően választva, saját stílust kialakítva oldja meg feladatait. Mégis gyakran hiányolunk néhány, máshol jól bevált utasítást, szerkezetet.

Megismerkedve néhány nyelvvel, engedtessék meg nekem, hogy közreadjam elképzeléseimet egy — legalábbis számomra — ideális nyelvről. Mint közgazdász-matematikus csakis felhasználói szempontból vizsgálom az általam ismert nyelvek néhány tulajdonságát és teszek maximalista javaslatokat. Mivel nem értek hozzá, ezért nem foglalkozom a megvalósítás lehetőségeivel és nehézségeivel, hatékonysággal, tárkapacitással, egyebekkel. Kant terminológiájával élve: a SOLLEN (aminek lenni kell) foglalkoztat, nem a SEIN (ami van).

Gondolataimat néhány jellemző köré csoportosítottam: ezek súlyát, egymáshoz való viszonyát a tárgyalás sorrendje nem jelzi.

Strukturáltság

A strukturált programozáshoz a szükséges gyakorlaton és önfegyelmen kívül bizonyos nyelvi elemek megléte elengedhetetlen, mások nélkülözhetőek ugyan, de nagyon hasznosak. Az IF—THEN—ELSE szerkezet, a REPEAT, WHILE, UNTIL ciklusszervező utasítások segítségével jórészt elkészíthető a programszöveg, ráadásul feleslegessé teszik a feltétel nélküli ugrás (GOTO) használatát. Nagyon fontos a lokális és globális változók, szubrutinok és függvények alkalmazhatósága. Gyakran hasznos valamely ciklus magjából való kilépésre az EXIT IF, különösen ha egymásba ágyazott ciklusok mélyéről is kiugorhatunk vele.

Adatstruktúrák

Legalábbis Pascal-szerű adatszerkezeteket — tömb, rekord, halmaz, fájl és ezek tetszőleges kombinációi — várok el, s lehetőséget saját adattípusaim megvalósítására (például gráfok, táblázatok stb. definiálása). Elvárom, hogy az értékadás és az összehasonlítás műveletei ezek egészére és részeire is lehetségesek legyenek, beleértve saját műveletek használatát is. Mélységesen elítélem az automatikus típusátalakításokat, viszont legyenek konverziós utasítások.

Interaktivitás

Ez a programok egészének, de főként az egyes részmoduloknak a belövéséhez, teszteléséhez az egyik leghatékonyabb lehetőség. Hiba esetén megvizsgálhatók a változók, módosítva őket, tovább futtatható vagy újraindítható a program. Nem véletlen az sem, hogy megannyi felhasználói programot alakítottak át interaktivvá.

Rekurzivitás

Bár minden problémát át lehet fogalmazni úgy, hogy ne legyen szükség rekurzió alkalmazására — néha persze rendkívül bonyolult lesz az eredménye —, mégis számos feladatot rekurzív algoritmussal a legegyszerűbb megoldani. Jellegzetes típusa a visszalépéses keresés, mint például a nyolcvezér-probléma. (A számtalanszor felhozott faktoriális számítás rossz példa, ugyanis az iteratív megközelítés ebben az esetben sokkal hatékonyabb.)

Lefordíthatóság

Sajnos az interaktivitás általában interpreter végrehajtást is jelent, s ez nagyon lelassítja a programok futását. Szerintem minden nyelvhez elvárható fordítóprogram, amellyel a belőtt, végleges programváltozatok futási ideje töredékére lenne csökkenthető.

Bővíthetőség

Eljutottam végre mondanivalóm lényegéhez. Az eddigieken túl olyan programnyelvet szeretnék használni, amely nyitott: kiegészíthető a célnak megfelelő új utasításokkal. S e bővítést ne gépi kódban, hanem az adott magas szintű nyelven készíthessem el, majd lefordítva, az utasításkészlethez csatolhassam! Így érhető el, hogy „feladatra" szabott nyelven íródjanak a programok, illetve bizonyos feladatosztályokhoz nagyon magas szintű nyelvek készüljenek.

Egyebek

A dekrementáló és inkrementáló utasításoknak és a pointerezett változókezelésnek szerepe lehet a bővítések gyorsasága szempontjából, de nem feltétlenül szükségesek. Furcsa, nagy lehetőségekkel és veszélyekkel kecsegtető módszer a karakterfüzérek programszöveggé alakítása, ill. a fordítottja, programrészek sztringgé konvertálása. Hasonló jellegű utasítás a sztringben megadott tetszőleges kifejezés kiértékelése, beleértve utasítások végrehajtását is.

Végezetül az igényeimhez közelálló nyelvekkel kapcsolatos kifogásaimat szeretném felsorolni.

APL — Nem strukturált, könnyű áttekinthetetlen s ezért javithatatlan programokat írni, nincs fordítóprogramja, nem lehet új adatszerkezeteket definiálni.

BASIC — Nem strukturált, nem rekurzív, nem bővíthető, mikrogépekre nincsenek jó fordítók. A Spectrum Beta BASIC-je és a QL Suberbasic-je már strukturált, de adatszerkezetei szegényesek.

FORTH — Túlságosan alacsony szintű és nehézkes, nagy fáradságba kerül akár egy BASIC szintjére is hozni.

LOGO — Sajnos kizárólag a listakezelésre épít, hiányzik belőle a többi adatszerkezet, nem lefordítható.

●

Mindezzel kapcsolatban kíváncsian várom az olvasók és a programozási nyelvekkel foglalkozó szakemberek véleményét.

LOVRICS LÁSZLÓ

Lovrics László egy, a programozási nyelvek története folyamán igen sokat vizsgált és vitatott kérdéskörhöz szólt hozzá. E körbe három, egymással összefüggő kérdéscsoport tartozik: a nyelvek *osztályozása, összehasonlítása* és *értékelése*.

Az első terjedelmes munka, amely ezzel foglalkozik, több szakirodalmi előzmény után jelent meg. (Jean E. Sammet: Programming Languages: History and Fundamentals. Prentice Hall Inc. 1969.) Fedőlapján Bábel tornyának stilizált rajza látható, az akkor volt programozási nyelvek neveivel teleírva. A téma azóta sincs kimeritve: 1984-ben például megjelent egy tanulmánykötet, amely Sammet átfogó művével ellentétben három nyelvre, az Adára, a C-re és a Pascalra összpontosít. (Alan Feuer—Narain Gehani [ed]: Comparing and Assessing Programming Languages Ada, C and Pascal. Prentice Hall Inc. 1984. Megjelenik magyar nyelven is, előreláthatólag 1988-ban, a Műszaki Könyvkiadó gondozásában.) A könyvben összegyűjtött tanulmányok a hetvenes és a korai nyolcvanas években jelentek meg, bizonyítva az e témakör iránti folyamatos érdeklődést. Ha megnézzük az ezekben és más helyeken publikált irodalomjegyzékeket, meggyőződhetünk róla, hogy e kérdéseknek mekkora irodalmuk van! Nincs szándékomban bővebben kifejteni e terület egyetlen aspektusát sem, de talán nem érdektelen, ha röviden összefoglalom a vizsgálódások irányait.

Az *osztályozás* kérdéséről megjegyzem, hogy igen sok szempont szerinti csoportosítás lehetséges, és szinte bármelyik esetén találhatók olyan nyelvek, amelyek valahogyan „kilógnak a sorból": vagy egyik kategóriába sem sorolhatók egyértelműen, vagy többe is besorolhatók. Ismeretesek osztályozások a nyelvek jellege szerint: általános célú (Pascal, PL/1.), specifikációs (SDLA), speciális feladattípusra orientált (LISP) s a többi; az utóbbi esetben alosztályok határozhatók meg a feladattípusok szerint. Egy másik osztályozás alapján megkülönböztethetők procedurális („Neumann-elvű") és nem procedurális (applikatív, deklaratív stb.) nyelvek. A legközismertebb osztályozás magas és alacsony szintű (gépközeli) nyelveket különböztet meg. Lehet a csoportosítás alapja az is, hogy milyen viszonyban vannak egymáshoz képest a nyelvben a tevékenységek és az adatok: az egyik vagy másik elsődleges, vagy egyenrangúak. Az osztályozási lehetőségek szinte kimeríthetetlenek.

Összehasonlítani akkor lehet két vagy több dolgot, ha azok valamilyen szempontból hasonlatosak. Ez a programozási nyelvekre is igaz. Ha két programozási nyelv valamilyen csoportosítási rendszerben különböző kategóriába tartozik, akkor — leg-

...Bábel tornyára?

alábbis az adott csoportosítás szempontjai alapján — nem hasonlíthatjuk őket össze, csupán a különbözőségüket rögzíthetjük.

Az összehasonlítás célja valamilyen értékítélet vagy döntés megalapozása, az összehasonlított nyelvek tulajdonságainak tüzetes vizsgálatával. Ha például a vizsgálat a numerikus mérnöki számításokra való alkalmazásra irányul, akkor ebből a szempontból összehasonlítható a FORTRAN és a Pascal, és feltehetők olyan kérdések, mint: melyik nyelv támogatja jobban a többszörös pontosságú aritmetikát, melyiknek gazdagabb a beépített függvény- és eljáráskészlete, melyikben kényelmesebb az eredmények táblázatba foglalása és kinyomtatása stb. Az összehasonlítás konklúziója egy olyan értékítélet vagy döntés, amely azt mondja ki, hogy az adott feladatosztályra — bizonyos kompromisszumokkal — valamelyik nyelv jobban vagy kevésbé ajánlható a másiknál. A következtetés tehát nem az, hogy az egyik nyelv „jobb", a másik „rosszabb".

Minthogy az összehasonlítás csak egy adott osztályozási rendszer adott kategóriáján belül jogos, egyik nyelvre tett kijelentések sem abszolút érvényűek. Ha egy alkalmazási területen vagy feladatban egy- és kétindexes változókon kívül nincs szükség egyébre, a még oly hajlékony adatstrukturálási lehetőségek sem jelentenek többletet, egy gráfbejárási feladatnál pedig a komplex aritmetika lehet teljesen értéktelen.

Az *összehasonlítástól* eltérő értelemben használjuk az *értékelés* kifejezést. Míg az összehasonlítás esetében két vagy több nyelv jellemzőit vetjük egybe, az értékelésnél egyetlen nyelv sajátosságait vesszük górcső alá. Minden nyelvnek van valamilyen alapfilozófiája — mondhatni: „alapítási okmánya" —, amely meghatározza, hogy hol a nyelv helye a nap alatt. (Amelyik nyelvnek ilyen nincs, azzal nem érdemes foglalkozni.) Néha a nyelv fejlesztői szabatosan leírják ezt a filozófiát, néha pedig a filozófia magából a nyelvből derül ki. (Szilárd elvi megalapozásról természetesen csak az első esetben beszélhetünk.)

Az értékelés első fázisában a nyelv filozófiáját kritizáljuk. Van-e rá szükség, vagyis hézagpótló-e a maga nemében, vagy csupán utánérzése valami korábbinak? Milyen tulajdonságokat hordoz? Túl sokat markol-e, vagy éppenséggel túl keveset? A meglévő osztályozási szempontok szerint be tudjuk-e sorolni valahová, és ha igen, hová? Egy későbbi összehasonlító elemzésnél milyen más nyelveket kell majd figyelembe venni?

A második fázisban a nyelvet szembesítjük a saját filozófiájával. Itt mindenekelőtt azt kell megnézni, hogy a nyelvi konstrukciók (vezérlési- és adatstruktúrák, utasításfajták, „támadó" és „védekező" mechanizmusok stb.) mennyire vannak összhangban a meghirdetett célokkal: elégségesek-e megvalósításukhoz, avagy többet, esetleg kevesebbet nyújtanak-e? Itt ismét hangsúlyozni kell a kijelentéseink érvényének viszonylagosságát: azok csak a nyelv filozófiája által meghatározott követelményrendszeren belül érvényesek. Túlzottan általánosít például egy olyan kijelentés, amely szerint a címaritmetika hasznos (vagy káros). A tényszerű megállapítás az lehet, hogy ez az adott nyelv céljával, rendeltetésével, egyéb nyelvi konstrukcióival stb. összhangban van (vagy nincs összhangban).

Vannak mindenfajta sajátosság fölött álló szempontok is. Például a logikai tisztaság, következetesség és szabatosság követelménye. Bármi legyen is egy nyelv célja, ha szerkezete zavaros, következetlen, ha a leírásban nem lehet eligazodni, ha a programok betűrejtvényekre emlékeztetnek, kín vele dolgozni. A zavarosság, következetlenség mögött gyakran tetten érhető valamiféle „fejlődési rendellenesség": egy korábban tiszta és világos filozófiájú nyelv továbbfejlesztése során a szerzők újabb, a korábbiakkal nem harmonizáló elemeket vezettek be, megbontva ezzel a konstrukció egyensúlyát. Jól nyomon követhető ez például a BASIC nyelv fejlődésén.

Szintén az értékelés kategóriájába tartozó, de az imént leírtaktól mégis elhatárolódó fogalom egy nyelv *adott gépi megvalósításának értékelése*. Míg az előbbi esetben a nyelv filozófiáját értékeltük, és a nyelvet szembesítettük a saját filozófiájával, itt egy adott megvalósítást állítunk szembe a hivatalos dokumentumokban vagy szabványokban meghatározott „etalon nyelvvel". Szemügyre vesszük továbbá a megvalósítás jellegét (értelmezőprogram, fordítóprogram, firmware stb.), annak hatékonyságát, a környezethez (operációs rendszer és segédprogramjai) való viszonyát, a felhasználói kényelmet stb.

Egy nyelv konkrét megvalósításának sajátosságai nem azonosak a nyelv sajátosságaival, bár kétségtelen, hogy a felhasználó hajlamos arra, hogy az előbbit azonosítsa az utóbbival. Gyakran hallott megállapítás, hogy a BASIC interaktív nyelv. Ez így természetesen nem pontos; a BASIC egyik megvalósítása lehet interaktív, egy másik pedig esetleg nem az. Ugyancsak gyakori, hogy a felhasználó valamelyik konkrét megvalósítással kapcsolatos kifogásait magára a nyelvre vetíti ki, mondván, hogy az X nyelv gyakorlatilag használhatatlan, holott valójában csupán az X nyelv valamelyik fordítóprogramja, futtató rendszere, felhasználói felülete vagy megvalósításának valamelyik egyéb tartozéka hagy maga után kívánnivalókat.

A megvalósítás sajátosságai és a nyelv sajátosságai mégsem teljesen függetlenek. A nyelv szerzői ugyanis a tervezéskor a megvalósítás szempontjait is szem előtt tartják. (Nagyon nagy baj, ha ezt nem teszik!) Ennek azonban az is követelménye lehet, hogy a nyelv definíciójába olyan, a nyelv filozófiájából egyébként nem levezethető sajátosságok is bekerülnek, amelyeket kizárólag valamilyen megvalósítási szempont indokol. A PL/1 nyelv szerkezetein ez több helyen is észrevehető. A (számítástechnika-történetileg nem teljesen igazolt) legenda szerint a FORTRAN-ban azért lehet éppen három indexe egy indexes változónak, mert az egykori IBM 704-es számítógépnek pontosan három indexregisztere volt.

Az összehasonlítás vagy értékelés, ha a szakszerűség és objektivitás igényével készült, tényeken kell hogy alapuljon. A konklúziója mégsem megdönthetetlen, hiszen a tények csoportosítása és súlyozása nem nélkülözheti a szubjektív elemeket. Ezért nem ritkák a homlokegyenest ellenkező végkövetkeztetések sem. Ugyanez igaz azokra a döntésekre is, amelyeket az ilyen elemzések alapján hozunk egy alkalmazási feladathoz megfelelő programozási nyelv megválasztásánál. Arra, hogy egy ilyen döntés mennyire szuverén, és mennyiben diktálják figyelmen kívül nem hagyható külső feltételek, itt nem lenne célszerű kitérni.

Programozási nyelvek osztályozására, összehasonlítására, értékelésére, konkrét megvalósítások feletti vitákra szükség volt a múltban is, és szükség van ma is. Ahhoz azonban, hogy ezekből akár vitatható következtetéseket is le lehessen vonni, mindig ki kell tűzni az elemzés célját és szempontjait. Ha ez elmarad, az érvek nemigen tükrözhetnek többet az érvelő egyéni ízlésénél. Az „ideális nyelv" kék madarát sohasem fogjuk utolérni. Keressünk inkább egy tűrhetően tojó kendermagos tyúkot!

LŐCS GYULA

Vigyázat! Tolvaj!

Az adatlevédési technikák további elterjedt módja az ellenőrző összeg (cheksum) megváltoztatása. A márciusi számban már ismertettük, hogy az FDC egy adatblokk lemezre írása esetén mind a blokkfejben, mind az adatblokk végén elhelyez egy-egy ellenőrző számot (lásd az *ábrát*), amelyek által meggyőződhet, hogy sikerült-e pontosan elolvasnia a lemezen lévő adatokat. Tulajdonképpen ez a második biztonsági ellenőrzés, mivel a GCR kód is hasonló szerepet tölt be.

A blokkfej felépítése

Az adatblokk felépítése

Az ellenőrzés folyamatában az FDC miután egy blokkot elolvasott, vagyis betöltötte a meghajtó egyik pufferébe, a GCR konverzió újra kiszámítja az ellenőrző számot, és ezt hasonlítja össze a lemezen lévővel. Ha ez különbséget mutat, akkor hibajelzéssel tér vissza, egyébként az adatokat továbbítja a soros buszon a C64-nek.

Láthatjuk, hogy az FDC a hibát csak az adatok pufferbe olvasása után ellenőrzi, tehát megtalálhatók a meghajtó memóriájában, de hiba detektálásakor ezeket nem továbbítja. Ha feladjuk ezt a második kontrollrendszert, amivel az olvasás csak kis mértékben válik bizonytalanná, akkor megtehetjük , hogy az ellenőrző összeget módosítjuk, vagyis átírjuk. Így olvasáskor hibajelzést kapunk ugyan, de egy segédprogrammal közvetlenül kiolvashatjuk a meghajtó pufferéből a már felhasznált adatokat.

Programunkat úgy érdemes elkészíteni, hogy legyen egy kis behívóprogram (boot), ezzel aktivizáljuk a főprogramot, amiben pedig elhelyezünk egy programrészt annak az egy-két blokknak a beolvasására, amelyekben módosítottuk az ellenőrző összegét. Természetesen ennek a néhány blokknak olyan fontos információt kell tartalmaznia, amely nélkülözhetetlen a program eredményes futásához.

A legtöbb másolóprogram ellen hatásos ez a levédési módszer, mert a másolók zöme általában hibás blokk esetén csak a hiba típusát veszi figyelembe. Egy hasonlót generál az új lemezre is, de a hibás blokk adatait már nem másolja át. Ilyen például a B—52 és a Cracker Copy is.

Most pedig nézzük, hogy milyen módon állítja elő a rendszer az ellenőrző összeget! A blokkfej-ellenőrző összeg a blokkfej adatainak kontrollja. Nem más, mint a szektorszám, a sávszám, az azonosító felső és alsó bájtja között elvégzett kizáró VAGY (EOR azaz Exclusive—ORing) művelet elvégzése után kapott szám. Egy blokk felírásánál az ellenőrző összeg kiszámítása után kerül sor a GCR konverzióra, majd az adatokat így tárolja a rendszer a lemezen. Az adatblokk ellenőrző összegét hasonló módon kell kiszámítani, csak a blokkfej-információ helyett az adatok között kell elvégezni az EOR műveletet, majd a GCR konverziót.

Nézzünk egy példát a blokkfej ellenőrző számának a kiszámítására a következő adatokkal:

	Hexadecimális bináris számrendszer
Szektorszám	$00 (0) 00000000
Sávszám	$12 (18) 00010010
ID LO	$58 (88) 01011000
ID HI	$5a (90) 01011010

1. lépés: EOR $00 (0) a szektorszámmal

$00 = 00000000
szektorszám ($00) = 00000000
00000000

2. lépés: EOR a sávszámmal

00000000
sávszám ($12) = 00010010
00010010

3. lépés: EOR az ID LO-val

00010010
ID LO ($58) = 01011000
01001010

4. lépés: EOR az ID HI-vel

01001010
ID HI ($5A) = 01011010
00010000

5. lépés: Bináris-hexadecimális konverzió

00010000 = $10 (16)

Ezután következik a GCR konverzió, majd a lemezre írás.

Most pedig nézzük a DOS Inside alapján készült programot, amellyel egy teljes sávban meg tudjuk változtatni a blokkfej-ellenőrző összeget. A program lefuttatása után az illető sáv blokkjainak olvasása esetén 27-es típusú hibaüzenetet kapunk.

KOVÁCS P. ATTILA

```
100 REM 27M ERROR - 1541
110 DIMD$(25)
120 PRINT"{CLR}MULTIPLE 27 ERROR - 1541"

130 PRINT"{DOWN}INSERT CLONE IN DRIVE"
140 INPUT"{DOWN}DESTROY TRACK";T
150 IFT<1ORT>35THENEND
160 INPUT"{DOWN}ARE YOU SURE  Y{LEFT 3}"
;O$
170 IFO$<>"Y"THENEND
180 OPEN15,8,15
190 PRINT#15,"I0"
200 INPUT#15,EN$,EM$,ET$,ES$
210 IFEN$="00"GOTO260
220 PRINT"{DOWN}"EN$","EM$","ET$","ES$
230 CLOSE15
240 END
250 REM SEEK
260 NS=20+2*(T>17)+(T>24)+(T>30)
270 S=NS
280 JOB=176
290 GOSUB580
300 FORI=0TO23
310 READD
320 D$=D$+CHR$(D)
330 I$=I$+CHR$(0)
340 NEXTI
350 PRINT#15,"M-W"CHR$(0)CHR$(6)CHR$(24)
D$
360 REM EXECUTE
370 PRINT"{DOWN}{RVS}DESTROYING{ROFF} TR
ACK";T
380 JOB=224
390 GOSUB580
400 PRINT#15,"M-W"CHR$(0)CHR$(6)CHR$(24)
I$
410 FORJ=0TO25
420 FORI=0TO7
430 READD
440 D$(J)=D$(J)+CHR$(D)
450 NEXTI
460 NEXTJ
470 I=0
480 FORJ=0TO25
490 PRINT#15,"M-W"CHR$(I)CHR$(4)CHR$(8)D
$(J)
500 I=I+8
510 NEXTJ
520 REM EXECUTE
530 PRINT#15,"M-E"CHR$(0)CHR$(4)
540 CLOSE15
550 PRINT"{DOWN}DONE!"
560 END
570 REM JOB QUEUE
580 TRY=0
590 PRINT#15,"M-W"CHR$(12)CHR$(0)CHR$(2)
CHR$(T)CHR$(S)
600 PRINT#15,"M-W"CHR$(3)CHR$(0)CHR$(1)C
HR$(JOB)
610 TRY=TRY+1
620 PRINT#15,"M-R"CHR$(3)CHR$(0)
630 GET#15,E$
640 IFE$=""THENE$=CHR$(0)
650 E=ASC(E$)
660 IFTRY=500GOTO690
670 IFE>127GOTO610
680 RETURN
690 CLOSE15
700 PRINT"{DOWN}{RVS}FAILED{ROFF}"
710 END
720 REM 21 ERROR
730 DATA  32,163,253,169,  85,141,  1, 28

740 DATA 162,255,160, 48, 32,201,253, 32

750 DATA   0,254,169,  1, 76,105,249,234

760 REM 27M ERROR
770 DATA169,  0,133,127,166, 12,134, 81
780 DATA134,128,166, 13,232,134, 67,169
790 DATA  1,141, 32,  6,169,  8,141, 38
800 DATA  6,169,  0,141, 40,  6, 32,  0
810 DATA193,162,  0,169,  8,157,  0,  3
820 DATA232,232,173, 40,  6,157,  0,  3
830 DATA232,165, 81,157,  0,  3,232,169
840 DATA  0,157,  0,  3,232,157,  0,  3
850 DATA232,169, 15,157,  0,  3,232,157
860 DATA  0,  3,232,169,  0, 93,250,  2
870 DATA 93,251,  2, 93,252,  2, 93,253
880 DATA  2,157,249,  3,254,249,  2,238
890 DATA 40,  6,173, 40,  6,197, 67,208
900 DATA186,138, 72,169, 75,141,  0,  3
910 DATA162,  1,138,157,  0,  5,232,208
920 DATA250,169,  0,133, 48,169,  3,133
930 DATA 49, 32, 48,254,104,168,136, 32
940 DATA229,253, 32,245,253,169,  5,133
950 DATA 49, 32,233,245,133, 58, 32,143
960 DATA247,169, 35,133, 81,169,169,141
970 DATA  0,  6,169,  5,141,  1,  6,169
980 DATA133,141,  2,  6,169, 49,141,  3
990 DATA  6,169, 76,141,  4,  6,169,170
1000 DATA141,  5,  6,169,252,141,  6,  6

1010 DATA169,224,133,  3,165,  3, 48,252

1020 DATA 76,148,193,234,234,234,234,234
```

```
170 ;
180 *= $0400
190 ;
200 ;* INITIALIZATION *
210 ;
220 LDA #$00
230 STA $7F
240 LDX $0C
250 STX $51
260 STX $B0
270 LDX $0D
280 INX
290 STX $43
300 LDA #$01
310 STA $0620
320 LDA #$08          ; TAIL GAP
330 STA $0626
340 LDA #$00
350 STA $0628         ; SECTOR COUNTER
360 ;
370 JSR $C100         ; LED ON
380 ;
390 ;* CREATE HEADERS *
400 ;
410 LDX #$00
420 HEADER # LDA #$08 ; HBID
430 STA $0300,X
440 INX
450 INX               ; CHECKSUM
460 LDA $0628
470 STA $0300,X       ; SECTOR
480 INX
490 LDA $51
500 STA $0300,X       ; TRACK
510 INX
520 LDA #$00
530 STA $0300,X       ; IDL
540 INX
```

Mesterséges értelem IV.

```
550 STA $0300,X      ; IDH
560 INX
570 LDA #$0F
580 STA $0300,X      ; GAP
590 INX
600 STA $0300,X      ; GAP
610 INX
620 ;
630 LDA #$00         ; COMPUTE CHECKSUM
640 EOR $02FA,X
650 EOR $02FB,X
660 EOR $02FC,X
670 EOR $02FD,X
680 STA $02F9,X
690 ;
700 INC $02F9,X      ; INCREMENT CHECKSUM

710 ;
720 INC $0628
730 LDA $0628
740 CMP #43
750 BNE HEADER
760 ;
770 TXA
780 PHA
790 ;
800 ;* CREATE DATA *
810 ;
820 LDA #$4B         ; 1541 FORMAT
830 STA $0500
840 LDX #$01         ; 1541 FORMAT
850 TXA
860 DATA STA $0500,X
870 INX
880 BNE DATA
890 ;
900 ;* CONVERT TO GCR *
910 ;
920 LDA #$00
930 STA $30
940 LDA #$03
950 STA $31
960 JSR $FE30
970 PLA
980 TAY
990 DEY
1000 JSR $FDE5
1010 JSR $FDF5
1020 LDA #$05
1030 STA $31
1040 JSR $F5E9
1050 STA $3A
1060 JSR $F78F
1070 ;
1080 ;* JUMP INSTRUCTION *
1090 ;
1100 LDA #$23
1110 STA $51
1120 ;
1130 LDA #$A9
1140 STA $0600
1150 LDA #$05
1160 STA $0601
1170 LDA #$85
1180 STA $0602
1190 LDA #$31
1200 STA $0603
1210 LDA #$4C
1220 STA $0604
1230 LDA #$AA
1240 STA $0605
1250 LDA #$FC
1260 STA $0606
1270 ;
1280 LDA #$E0
1290 STA $03
1300 ;
1310 WAIT LDA $03
1320 BMI WAIT
1330 ;
1340 JMP $C194
```

Bevezetés

„A számítógépek csak azt tudják megcsinálni, amiről mi előzőleg elmondjuk nekik, hogyan csinálják." Ezt a kijelentést gyakran szokták — tévesen — úgy értelmezni, hogy a számítógépek képtelenek tanulásra vagy új ismeretek felfedezésére. Pedig ezt így egyáltalán nem állítja, hanem mindössze annyit jelent csak, hogy nekünk kell jól megfogalmaznunk, miképpen tanuljon a gép, vagy hogy milyen módszerek vezetnek új összefüggések felismeréséhez. Mivel saját tanulási mechanizmusunkról sincs pontos elképzelésünk, érthetően nehézséget okoz meghatároznunk a számítógép számára a szükséges algoritmusokat.

Az emberek az agyukat alkotó neuronhálózat struktúrájának változtatásával tanulnak. Kezdetben a kutatók sok véletlen hálózatos kísérletet végeztek például perceptronokkal, de nem sok sikerrel. A kudarcok egyik oka az lehet, hogy amikor az emberek megtanulnak valamit, akkor a bonyolult új ismeretet általában a már rendelkezésre álló valamivel kevésbé bonyolult elemekből építik fel. Ez a folyamat egészen kis gyermekkortól kezdődően — jó esetben hosszú évtizedekig — tart. Ha egy véletlen hálózattal nulláról indulunk, nem is juthatunk egyhamar túl messzire. Ezért került — néhány évnyi egy helyben topogás után (főként P. H. Winston eredményei alapján) — e téren is a felületről történő megközelítés előtérbe.

A tanulás négy nézőpontból

A tanulás legegyszerűbb és legáltalánosabb — majd mindent tartalmazó, ezért nem is túl sokat mondó — definíciója az lehetne, hogy tanulás minden, ami által vagy melynek során a tanuló — vagy általánosabban fogalmazva: a rendszer — valamilyen értelemben a teljesítményét növeli.

Különösen az úgynevezett szakértői rendszerek elterjedése óta (v. ö. sorozatunk II. részével) vált népszerűvé az a definíció, miszerint a tanulás egy olyan folyamat, melynek eredményeként a rendszeren belül explicit módon kifejezett információ halmozódik fel.

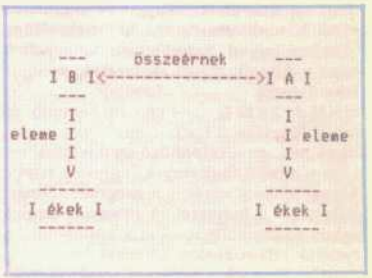
A harmadik (inkább pszichológiai jellegű) meghatározás szerint a tanulás készségek szerzése. Ezt azzal szokás alátámasztani, hogy miután valaki utasításokat kap egy feladat végrehajtásának módjára, a feladat ismétlésekor nyújtott teljesítményében — egy bizonyos határig — mégis fokozatos növekedés figyelhető meg. A jelenség feltehetőleg a feladat végrehajtásában közreműködő idegpálya-kapcsolódások (szinapszisok) megerősödéséből adódik.

A negyedik, legnehezebben megközelíthető és a kreativitással is szoros kapcsolatban álló tanulásmeghatározás szerint a tanulás az elképzelések, feltevések formálása, az összefüggések megtalálása.

A mostani tárgyalási kereteink jóval szűkösebbek annál, hogy ezt a szinte végeláthatatlan témát a felületnél akár csak egy kicsit is mélyebben tekintsük át, ezért az eddigiekhez hasonlóan megpróbálok a gyakorlati példák körében maradni.

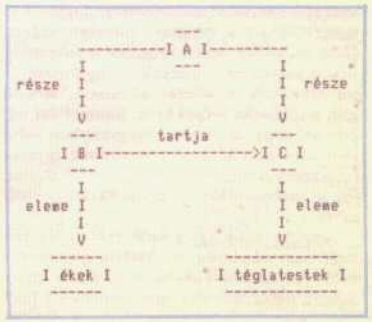
Gépies ismeretgyűjtés

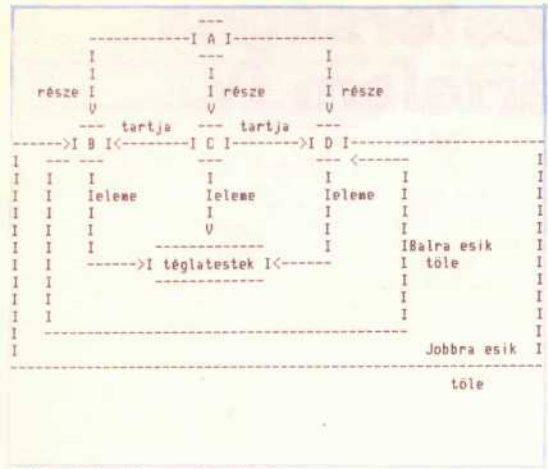
Ha az adatok gyűjtését és tárolását tanulásnak tekintenénk, akkor majd minden számítógépprogramot tanulónak nevezhetnénk. Ez a képtelenség önmagában indirekt bizonyítja — vagy ha tetszik, illusztrálja —, hogy a tanulás nem csak adatgyűjtést jelent. Mindazonáltal található az adatgyűjtésnek egy olyan formája, ami bizonyos fokig mégis egyszerű tanulásnak tekinthető. Erre példa, amikor egy játékprogram azokhoz a táblaállásokhoz, amelyekkel már találkozott, tárolja a kiértékelő függvény eredményét, esetlegesen a játszma végkimenetelét is adott lépés esetén. Ily módon fokozatosan növekszik majd a teljesítménye, mert hasznosítja a „tapasztalatait". Érezhető, hogy a sikerhez — ehhez a tanulási módszerhez — mennyire fontos (kritikus!) a tárolás körültekintő rendszerezése és (főleg!) a visszakeresés nagyon ügyes megszervezése.



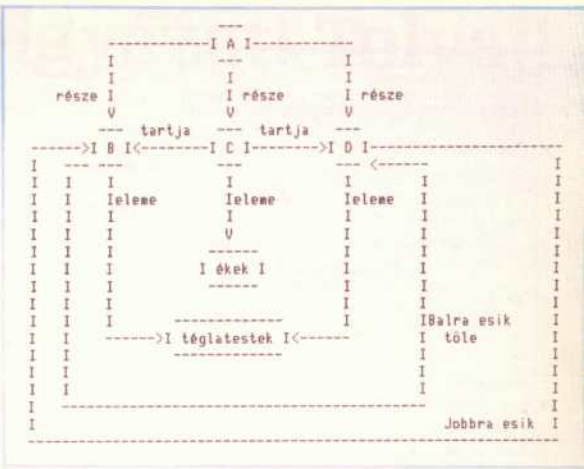
1. ábra. Sátor

2. ábra. „Ház" strukturális leírása





3. ábra. A „kapu" koncepció egy leírása



4. ábra. Az alternatív „kapu" koncepció leírása

Az ilyen adatgyűjtő tanulásnak igazából csak az olyan jellegű feladatoknál kellene szerepet játszania, ahol a környezeti információ direkt módon használható fel, elemzés szükségessége nélkül. Közismert szituáció a legjobb példa erre: a rendhagyó esetek megjegyzése nyelvtanulás során.

Tanulás paraméter-beállítással

A szakirodalom erre Samuel programját szokta klasszikus példaként említeni, mely a dámajátékra készült 1963-ban (4). A program alapját egy egyszerű minimax algoritmus alkotta, amely a táblaállásokat (az ilyen típusú feladatoknál szokásos módon) statikus értékelőfüggvény alkalmazásával hasonlította össze. Az értékelőfüggvényben olyan szempontok szerepeltek, mint a figurák száma, helyzetük, mozgási lehetőségeik. Ezekből a $c1 \times t1 + c2 \times t2 + \ldots + cn \times tn$ formájú értékelőfüggvény (c1, c2,. . .cn — súlyozófaktorok, t1. . .tn a különböző szempontok szerinti minősítő függvények) egyetlen értéket képezett, mely végül is a program számára jellemezte a helyzetet. A program megírásakor a súlyozófaktorokat a körülbelül helyesnek vélt értékekre állították be.

Itt gyökerezik a program sorsának érdekessége a témánk szempontjából. Ugyanis a program saját játéktapasztalatai alapján ezeket a súlyokat módosította, vagyis a finombeállítást a program önmaga végezte el. A működés során azoknak a jellemzőknek a súlyozása, amelyek — úgy tűnik — jól tükrözték a sikeres állásokat, megnő, míg a többiekét lecsökkent. Samuel ezt úgy érte el, hogy az értékelőfüggvényben sohasem szerepelt az összes szempont egyszerre, hanem a függvényben bentlévőket — és így a módosulókat — periodikusan cserélgette.

Nem merült ki így a problémák sora, maradt más nehézség is. Melyik paramétert módosítsuk és mikor, ha igen, akkor menynyivel, no és: mindet ugyanannyival? Ezek nem könnyű, de lényeges kérdések. Ráadásul az egyes játéklépésekről nincs visszajelzés, csak egyetlen egyszer: a játszma legvégén.

A paraméterek finombeállításához szükséges nagyszámú játszma lebonyolítását úgy szimulálták, hogy a program két másolata játszott egymás ellen oly módon, hogy mindig csak egyikük volt tanuló üzemmódban. Természetesen a játszmák végén a két program értékelőfüggvénye eltért egymástól, hiszen a tanuló üzemmódban levő program a tapasztalatok szerint a sajátját módosította. A partik után a vesztes programokat törölték, és a győztes két példánya játszott tovább — mint említettük, az egyik hátrányos pozícióban. Ha visszagondolunk sorozatunk első részére, akkor észrevehetjük, hogy a hegymászó stratégia egy változatával van dolgunk.

Tanulás példák alapján

Egy másik, az előzőnél az ember számára mindenképpen szimpatikusabb (emberközelibb) lehetőség arra, hogy tanítsunk: példákat mutatunk, hogy miként kellene viselkedni.

A rendszernek aztán ezeket az ismereteket általánosítania kell, hogy ily módon magasabb szintű szabályok segítségével a teljesítményét fokozhassa. Például, hogy a két mellső végtag felemelése esetén nemcsak a gazdi, hanem „mindenki", azaz ő (a kutya) is kap cukrot. Az általánosítás során természetesen vigyázni kell, mert túlozni itt sem szabad! (A hátsó láb felemelése nem feltétlenül jutalmazott cselekedet, különösen nem perzsaszőnyegen . . .)

Amikor bonyolultabb az általánosítás, ezt jellegzetes esetek citálásával segíthetjük. Egy osztályozási feladaton szeretném a dolgot bemutatni.

Az *1., 2. ábrák* két egyszerű objektum, a ház és a sátor reprezentálását mutatják be. Tegyük fel, hogy a már ezeket fölismerő programot ezután a Kapu fogalom megkülönböztetésére szeretnénk megtanítani. Kövessük végig, amint példákból általánosítva

„Az idomítás csodája"

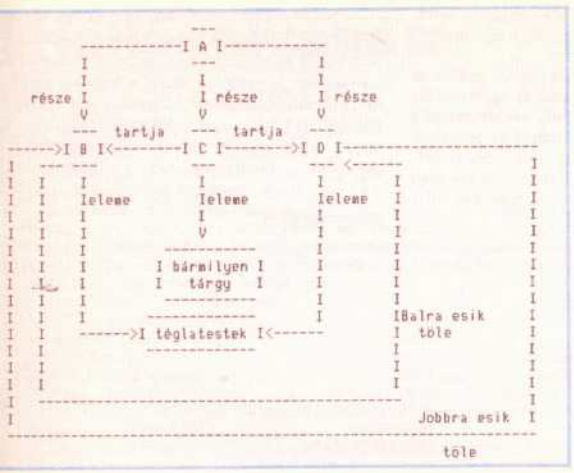
kialakít egy megfelelőnek látszó kapu értelmezést, majd a téves egyezéseket kiszűrve eljut a fogalom pontos általánosításához.

„Általánosabb ház koncepció"

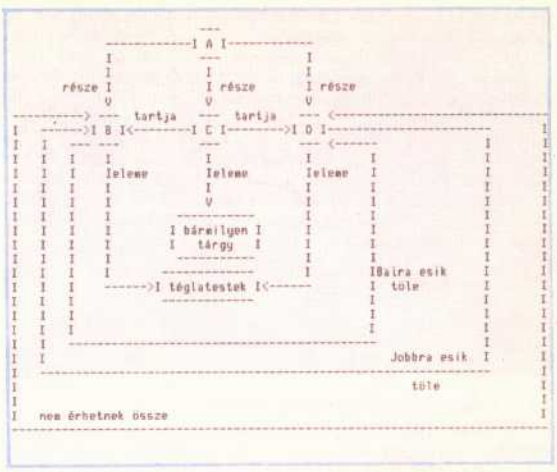
A program algoritmusa a következő:

1. Vedd a fogalom esetét. Ezt nevezd ki a fogalom definíciójának.

2. Vizsgáld végig a fogalom többi példáját. Bővítsd ki a definíciót oly módon, hogy ezekre is érvényes legyen.



5. ábra. A kapu definíció az általánosító lépés után



7. ábra. A Kapu fogalom végleges, megfelelően leszűkített definíciója

3. Vizsgáld végig a felsorolt közeli tévedéseket. Korlátozd úgy a definíciót, hogy ezeketmár zárja ki.

A közeliség (a nem nagy különbségek számbavétele) azért lényeges, hogy a hibák „megfoghatók", „instruktívak" legyenek.

Ezek alapján a program először elfogadja a *3. ábrán* látható kapuleírást a kapu fogalom definíciójának. Továbbhaladva a második lépésre, a *4. ábra* kapu példáját kezdi vizsgálni. A különbségeket elemezve felfedezi, hogy az oldalak által tartott test az ékek osztályába is tartozhat. Ezután feltételezi, hogy akkor a tartott tárgy helyén bármilyen objektum szerepelhet. A Kapu koncepció ekkor érvényes definíciója az *5. ábrán* látható. Ezután megvizsgálja a *6. ábrán* látható „közeli tévedés" példákat.

A kapura vonatkozó példából látszik, hogy a kapu két oldala nem érhet össze, hiszen lényegét veszti el az objektum, nem lesz többé kapu. Ennek alapján a programnak le kell szűkítenie a kapu definícióját oly módon, hogy már ne tartalmazza az olyan objektumokat, amelyeken nem lehet átmenni. A Kapu végleges definíciója a *7. ábrán* látható.

Bár az algoritmus elvi működését áttekintettük, még számos megoldandó kérdés maradt a programon belül. Ilyen például, hogy a programban valahol meg kell adnunk a nélkülözhetetlen tudnivalókat, vagyis szaknyelven: reprezentálva kell legyenek az össze nem érési tulajdonsággal kapcsolatos ismeretek.

A hasonló, példákból tanuló programok között akad, amelyik az oktatási példasorozat gondos összeállítására érzékeny. Más programok, mint például az itt említett is, az igen közeli tévedések bemutatására támaszkodnak.

Analógiás tanulás

Az analógia, ha úgy nézzük, igen különös és érdekes információátadási mód. Annak a mondatnak, hogy „Pistike olyan, mint egy tűzoltóautó" nyilvánvalóan van információtartalma, de ennek kihámozása már annak a feladata, aki a mondatot hallotta és kíváncsi Pistikére. (Kíváncsi lehet a tűzoltóautóra is, de ennek nagyságrendekkel kisebb a valószínűsége.) Neki kell kitalálnia, hogy vajon mi is az, amiben Pistike hasonlítható a tűzoltóautóhoz.

Winston ír le egy olyan programot, amelyben az analógiás következtetést tanulásra használja fel. A program ismeretei keretekbe vannak rendezve (lásd az előző részt). Ekkor az analógiás tanulás úgy képzelhető el, mint értékek átvitele az egyik keret valamelyik mezőjéből egy másik keret valamelyik mezőjébe. A *8. ábrán* látható példakeretek közül a TŰZOLTÓAUTÓ-keret a kiindulási, a PISTIKE pedig a célkeret. Az algoritmus lépései:

1. A kiindulási keret alapján a program kiválaszt mezőket, amelyek tartalma átvihető a célkeretbe. Ez a következő heurisztikák (lásd az első részben) alapján történhet:

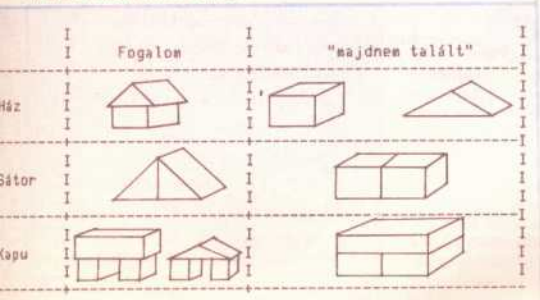
1.1. Válaszd azokat a mezőket, amelyek szélsőséges értékekkel vannak feltöltve. Ezt az magyarázza, hogy analógiát gyakran valamilyen különleges mértékben meglévő tulajdonság bemutatására alkalmazunk.

1.2. Válaszd azokat a mezőket, amelyekről már valahonnan tudod, hogy fontosak.

1.3. Válaszd azokat a mezőket, amelyek a kiindulási kerethez hasonló keretek között *nem szokásosak*. (Ez a lépés kifejezi: általában úgy választjuk, amivel párhuzamot vonunk, hogy azért lehetőleg elég világos legyen, amiért az adott dologhoz hasonlítunk . . . Tehát ez ismét abból a feltételezésből adódik, hogy valamilyen, az átlagtól eltérő tulajdonságra akarjuk felhívni a figyelmet.)

1.4. Válaszd azokat a mezőket, amelyek-

6. ábra. A közeli tévedés esetei



8. ábra. Három példa keret az analógiás tanulás magyarázatához

PISTIKE	ELEME	SZEMÉLYEK
	NEM	FIÚ
	AKTIVITÁS-SZINT	
	HANGOSSÁG	
	AGRESSZIVITÁS	KÖZEPES
TŰZOLTÓAUTÓ	ELEME	JÁRMŰVEK
	SZÍNE	PIROS
	AKTIVITÁS-SZINT	GYORS
	HANGOSSÁG	IGEN NAGY
	ÜZEMANYAG-KIHASZNÁLÁS	ÁTLAGOS
	LÉTRA-HOSSZ	XOR(LONG,SHORT)
AGRESSZIVITÁS	ELEME	SZEMÉLYISÉGJEGYEK

nek az értéke a kiindulási kerethez hasonló keretek között szokatlan, szélsőséges.

1.5. Végül használd fel a kiindulási keret összes mezejét. Lehetséges, hogy az analógia valamilyen rendkívüli egyezésre fogja felhívni a figyelmet.

2. Az előző lépésben összeszedett, felhasználhatónak látszó mezők közül most a célkeretben lévő információk alapján megpróbáljuk kiszűrni, hogy a javasolt analógiák — átvitelek — közül melyek használhatók. A kizárólag a kiindulási keret alapján javasolt mezők között most a további információk segítségével szelektálunk.

2.1. Válaszd azokat a mezőket, amelyek a célkeretben üresek. (Hiszen az analógia feltehetően a célkeret által reprezentált dologról kíván további információkat közölni.)

2.2. Válaszd azokat a mezőket, amelyek a célfogalom egy tipikus keretében jelen vannak.

2.3. Ha 2.2-nek semmilyen mező nem felelt meg, válaszd azokat, amelyek a célkerethez hasonló keretekben fellelhetők.

2.4. Ha még mindig nem felelt meg egyik mező sem, válaszd azokat, amelyekhez hasonló mezők a célkeretben vannak.

2.5. Végül válaszd azokat a mezőket, amelyekhez hasonló mezők vannak a célkerethez hasonló keretekben.

Látható, hogy fokozatosan egyre tágabb rokonságot, kapcsolatot engednek meg a felsorolt heurisztikák.

Kövessük a módszer alkalmazását Pistike és a tűzoltóautó konkrét példáján:

— A rutin a „Pistike olyan, mint egy tűzoltóautó!" állítás alapján indul.

— Ekkor a program előkeresi Pistike keretét és a tűzoltóautót reprezentáló keretet.

— A kiindulási — TŰZOLTÓAUTÓ — keretben az AKTIVITÁS és a HANGOSSÁG mezőkben található extrém érték, ezért ezeket az 1.1 heurisztika értelmében kiválasztjuk.

Ha ezek nem volnának jelen, akkor az 1.2. szabály nem választaná ki egyik mezőt sem, viszont az 1.3. szabály kiválasztaná a LÉTRA-HOSSZ mezőt, mivel ez semelyik másik JÁRMŰ-keretben nem fordul elő. 1.4. a SZÍN mezőt választaná, mivel igen kevés másik JÁRMŰ-keretben fordul elő a PIROS szín.

Érdemes megjegyezni, hogy ha az AKTIVITÁS és a HANGOSSÁG nem választódott volna ki a 2.2. szabály alapján, akkor a 2.4. szabály ismerte volna fel a bennük rejlő információtartalmat. Ugyanis az AGRESSZIVITÁS egy SZEMÉLYISÉGJEGY, és a szabály szerint más személyiségjegyek is kiválaszthatóak. Az AKTIVITÁS és a HANGOSSÁG pedig szintén személyiségjegyek.

PISTIKE	ELEME	SZEMÉLYEK
	NEM	FIU
	AKTIVITÁS-SZINT	GYORS
	HANGOSSÁG	IGEN NAGY
	AGRESSZIVITÁS	KÖZEPES

9. ábra. Az analógia segítségével szerzett ismeretekkel kiegészített Pistike keret

— Mivel tehát az AKTIVITÁS és a HANGOSSÁG mezőknek nem kétséges a kiválasztása, az algoritmus továbblép a második lépésre.

— Itt először a 2.1.-es szabályt próbálja alkalmazni. Ez egyetlen javaslatot sem utasít vagy enged el, ezért továbblép a 2.2. szabályra.

— A 2.2. szabály elfogadja az AKTIVITÁS és a HANGOSSÁG mezőket, mivel ezek jelen lehetnek egy tipikus SZEMÉLY keretben.

Tehát végül az analógia által kiegészített PISTIKE keret, amely most már tartalmazza az újonnan megtanult ismereteket is, a *9. ábrán* látható.

SOMOGYVÁRI KÁROLY

„Pistike egy sajnálatos programhiba óta . . ."

Ajánlott irodalom:

[1] Winston P. H.: The Psychology of Computer Vision: McGraw-Hill, New York, 1975.

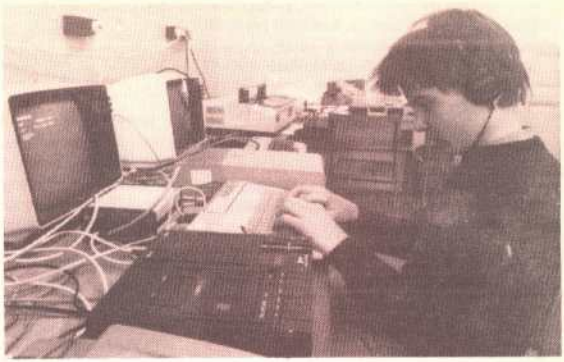
[2] Winston P. H.: Learning by Creating and Justifying Transfer Frames: az Artifical Intelligence an MIT Perspective c. könyvében, szerkesztők: Winston P. H., Brown R. H.; MIT Press, Cambridge, Mass., 1979.

[3] Cohen Paul R., Feigenbaum Edward A.: The Handbook of Artifical Intelligence, vol. 3.; William Kaufmann Inc., 1982.

[4] Samuel A. L.: Some Studies in Machine Learning Using the Game of Checkers: a Computers and Thought c. kötetben, szerkesztők: Feigenbaum E. A. és Feldman J., McGraw-Hill, New York, 1963.

Hallásolvasás

A Plus pluszt ad



Érzékeink, érzékszerveink a világgal örökös, kimerithetetlen mozgásban állnak. Így a beérkező információkat folyamatosan alakítjuk térben, időben összefüggően értelmezhető formába. Látásunk, hallásunk, szaglásunk, tapintásunk által érzékeljük a tárgyak mozgását, illetve helyhez kötöttségét.

Így történik ez az ép emberek világában. De mi van, ha valakinek valamely érzékszerve sérült? A többivel bár pótolni nem tudja, de képes a hiányos érzéklet kiegészítésére. Ennek ellenére nagyon nehéz fogyatékosnak lenni. A másságot gyakran fogadja idegenkedéssel az egészséges, még akkor is, ha különben segítőkész; és ez sokszor óhatatlanul szánalommal keveredik. Pedig a hátránya kerültek beilleszkedésének támogatásánál csupán esetlen maskara a szánalom.

Az a természetes, hogy mindenki teljes értékűen szeretne hozzájárulni a társadalom, szűkebb-tágabb környezete gyarapításához. A vakok is így éreznek, és ez a szándékuk mindinkább valós alapokra épülhet. A legkorszerűbb technika hasznosítását kiagyaló, alkotó ember ugyanis ezúttal az ő segítségükre sietett. Egy kis gép közreműködésével perspektivikusan szinte korlátlanul képessé tehetők mindarra, ami róluk korábban elképzelhetetlen volt.

Először talán hihetetlennek tűnik, hogy ma már nemcsak egy látó iskolás, hanem a vak kisgyermek is elkalandozhat a számítógépek birodalmában. Pedig így van. Működnek ugyanis olyan berendezések, amelyekkel a nem látó személy is programokat készíthet, tanulásához használhatja a BASIC nyelvet, éppúgy szórakozhat a számítógépen, mint tökéletes érzékszervű társai.

Arató András villamosmérnök és vak felesége, aki programozó, a KFKI-ban dolgoznak. A mérnököt az asszony helyzete motiválta a szóban forgó téma kutatásában. Holtzer Lóránt nak, a KFKI gazdasági igazgatójának támogatásával, a helyi KISZ-esekkel együtt még 1978-ban elkészítette az egykarakteres Braille-kijelzőt. Ma öt vak programozó dolgozik ilyen berendezésekkel; az intézet ajándékaként jutottak hozzá.

Arató a következő eszközt is feleségének csinálta: egy mikroprocesszoros fejlesztőrendszert, amely noha korlátozott szókészlettel beszélt, de már angolul szólalt meg. Folyományа ez az asszonnyal — Vaspöri Terézzel — kezdett nagyszabású vállalkozásának. Néhány esztendeje ugyanis együtt fáradoznak azon, hogy ne csak programozóknak készüljenek kisegítő berendezések, hanem más munkakörben dolgozó vagy más pályára készülő látásfogyatékosoknak is.

— Ezekkel a segédeszközökkel új munkalehetőségeket teremthetünk a vakoknak — mondja a férfi. — A KFKI kutatási témaként hagyta jóvá ez irányú munkánkat; az intézet mindenekelőtt emberiességi megfontolásból döntött így. Sajnos a gyors, hathatós munkához nem kapunk központi anyagi támogatást. Természetesen ez az olcsóbb előállítás rovására megy. Költségvetésünket leginkább adományokból fedezzük. Mostanáig két helyről érkezett nagyobb összeg: a Fővárosi Kefe- és Seprűgyártó Vállalattól és a Fregatt Sörözőben megrendezett „Zenehét a vakokért" adománygyűjtő dzsesszkoncertek bevételéből.

Több mint egy éve 30 BraiLab – ezek Homelab alapú számítógépek, amelyeket Lukács András és Endre fejlesztettek ki — működik már az országban. Ez az első magyarul szóló személyi számítógép. Kifejezetten vakoknak készítették a KFKI-ban. A látásfogyatékosok önállóan kezelik a beszélő BASIC interpreterét. Az okos, praktikus készülék alapfokú ismerkedést tesz lehetővé a számítástechnikával. A BraiLab-bal a Vakok és Gyengénlátók Országos Szövetségében több számítástechnikai ismeretet nyújtó, három hónapos alapfokú tanfolyamot indítottak. Akik némi jártasságra tettek szert a BraiLab kezelésében, azoknak kéthetente klubfoglalkozást tartanak.

Csák Valéria a szövetség egyik szervezője. Mint elmondta, olyan rendezvényeknek szeretnének otthont adni, ahol az értelmiségiek szórakozva, hasznosan töltik idejüket. A látásfogyatékosoknak a HCC klubtól szerzett és saját készítésű programokkal együtt már 15—20 játék- és szövegszerkesztő programjuk van. Így készültek el a telefonkönyves, a sakk- és (kuriózium ként) a fogyókúrás programok.

A BraiLab használatára a szerencsésebbeknek nem kell felnőtt korukig várniuk. Két ilyen készülék már ott van a vakok általános iskolájában, ahol iskolaszámítógépként működnek. Várhelyi Iván irányításával eddig történelem, tájékozódás, közlekedési ismeretek és megannyi játékos foglalkozás oktatóeszközéül szolgálnak a gépek. A tervek szerint ezek mellé még tíz másik sorakozik majd abból a hatvanból, amelyeket Dombóváron a Color Ipari Szövetkezet gyárt. A Vakok és Gyengénlátók Szövetségén kívül magánszemélyek is vásárolhatják a készülékeket, OTP-kölcsönre is. Egy BraiLab ára kb. 17 ezer forint lenne, de minthogy szociális célokat szolgál, a KFKI a licencdíjat elengedi a gyártónak, és így kétezer forinttal olcsóbbá teszi a forgalmazást.

Visszatérve a vakok általános iskolájához: ha megkapják az újabb tíz készüléket, akkor egy teljes felső tagozatos osztály szaktárgyai oktatásában lesznek integrálhatók a számítógépek. Heleszfai Katalin, az iskola igazgatónője nagy örömmel szól a BraiLabos tanulás lehetőségeiről:

— A segédeszköz élvezhetővé teszi a nehezen feldolgozható tantárgyakat. Módot ad rá, hogy a vak tanuló saját magát — pontos vagy hibás ismereteit — ellenőrizze, korrigálja. A tanításban rendkívül fontosnak tartjuk a BraiLab-park gyarapítását, mert ezzel a nem látó gyerekek is — a látó iskolásokhoz hasonlóan — más irányból is elmélyülhetnek a tananyagban.

Egy határozott nekirugaszkodás elvetélt kísérletének tekinthető-e a BraiLab, vagy kedvező fogadtatásra talált? Követi-e egy újabb számítógép fejlesztése? Mindez elsősorban Arató Andráson és munkatársain múlik.

— Egy olyan gépen dolgozunk, a BraiLab Pluson, amellyel szeretnénk elérni, hogy kedvező áron mind többen hozzájussanak korunk „varázsdobozához". Mit tud majd a berendezés? Nem igényel különösebb üzemeltetési környezetet, hajlékonylemezes háttértárolóval, beépített RAM lemezzel készül. Ez egy 256 kbájtos beszélő szerkezet, beszélő szövegszerkesztőt és beszélő adatbáziskezelőt építettünk bele. Fejlesztünk hozzá egy terminálemulátor programot is, amelynek révén a szabványos soros vonali illesztővel hozzáköthető majd az IBM PC-hez és nagyobb teljesítményű számítógépekhez.

Olyan programot is ki akarunk dolgozni, amely a számítógépből nyert információt a magyar Braille írás szabályait figyelembe véve nyomtatja. Ezenkívül például beválhat vak jogászok munkájának megkönnyítésére, jogi adatbázisok olvasásához. Vagyis a BraiLab Plus mintegy hídként vezeti majd a vakokat a számítógépes információk megszerzésében. Beszélő írógéppel hazánkban is megteremthető a látásfogyatékosok számára a leíró szakma. Ugyanis míg Nyugaton jól alkalmazzák gépíróként a látásfogyatékosokat, addig nálunk az ehhez szükséges berendezések hiányában ez a munkakör számukra csak vágyálom volt. De aki a BraiLab Plust jól ismeri, éppúgy lehet telexkezelő, mint diszpécser is. Igaz, a gépi hang a fülnek kissé torz, de rövid hallgatás után értelmezővé, megszokhatóvá válik. Mindenesetre körükben csodaszámba megy, hogy valaki visszajelzést kap — hangi élményként — arról, hogy mit és hogyan dolgozik.

A tervek szerint az év végére elkészül az első BraiLab Plus. De hogy a számítógép-alkalmazásban a rátermett vakoknak ne csak érdemi tudásuk legyen, hanem a környezet és a társadalom is a jelenleginél jobban elfogadja képzettségüket, munkahelyeket teremtsen részükre, ahhoz még gyökeres szemléletváltozásra van szükség a fejekben. Manapság már 8-10 vak programozó dolgozik. Jó példával járnak elöl: munkájukkal előkészítik a többieknek a jövő tágabb lehetőségeit.

KRASZNAI ÉVA

Az adattípu

A cikk első részében az absztrakt adattípusokról és a velük operáló programozási nyelvekről szóltunk. Most a fejlődés további történetét tekintjük át.

A típusfogalom finomodásában a következő lépés Zilles és Liskov nevéhez fűződik, és a CLU nyelvben, illetve Burstall által a CLEAR nyelvben lett hasznosítva, érvényesítve. Ők a típus fogalmát a matematika iniciális algebra fogalmával kapcsolták össze. (Ezt azért tették, mert algebrai úton akarták a programok helyességét bizonyítani, mi azonban csak az egyszerű programírók szempontjából vizsgáljuk a kérdést. Az eredmények így is érdekesek.) Az *algebrai indíttatású típusfogalmat* a következőkben foglalhatjuk össze: a típus nem más, mint egy függvénykészlet, és a nulla argumentumú függvényeket konstansoknak nevezzük. A típus azon értékek gyűjteménye, amelyek a konstansokból az adott függvények segítségével előállíthatók.

A gondolat alapján *a programot úgy bontjuk modulokra*, a CLU elnevezésével *clusterekra, hogy* minden cluster éppen egy, *a felhasználó* által igényelt adattípus konstansait és kezelő függvényeit tartalmazza. A cluster magába zárja azokat az információkat, hogy hogyan van a típus a számítógépen ábrázolva; ez a programozó magánügye, amelybe nem illik beleszólni. A felhasználó, aki a clustert programjába be akarja építeni, semmit sem használhat fel a belső ábrázolásból, *az adattípus adatait csak a definiált kapcsolatokon* (az adott függvényeken) *keresztül kezelheti*. Az így létrehozott program olyan jól modularizált, hogy bármelyik modulját kicserélhetjük egy másképpen megírt, de azonos kapcsolatúra az összes többi megzavarása nélkül.

Buktatók

Ennek a felfogásnak a gyenge pontja — ami a mai napig újra fellépő probléma — az úgynevezett vegyes típusú műveletek kapcsán jelentkezik. *A kérdés lényege, hogy a vegyes operandus típusú műveleteket melyik típushoz kapcsoljuk.* Vegyünk például egy egyszerű osztást: az út hosszát osztva az idővel kiszámíthatjuk az átlagsebességet. Vajon ez a függvény az út, az idő vagy a sebesség típushoz tartozik-e? Amellett, hogy ezt a műveletet nemigen lehet leírni a belső reprezentációk figyelembevétele nélkül, mert ha például a sebességhez kötjük, akkor a sebesség leírása függeni fog az út és az idő leírásától, azaz függőségi reláció alakul ki a típusok között.

Itt térhetünk ki az adattípusok elméletének egyik mellékvágányára. Vannak, akik *a típust a mértékegységgel azonosítják*. Ugyan a PASCAL óta többen mondták, hogy a méter és a liter két különböző típus, és ezért értékül nem adhatók egymásnak és össze sem adhatók, de az állítás igazi konzekvenciáit nagyon kevesen gondolták végig. Az egyszerű számokkal (programírói műszóval élve a skalárokkal) végzett műveletek alapvetően két csoportba oszthatók: additív és multiplikatív műveletekre. Mindenki előtt világos, hogy összeadni és kivonni csak azonos nemű dolgokat lehet, de a szorzásra és osztásra ez nem áll. Míg egyrészt tudható, hogy két lebegőpontos szám szorzata lebegőpontos lesz, másrészt az is biztos, hogy méter × méter × méter nem méter lesz, hanem köbméter, amelynek a literhez viszont egyértelmű köze van.

Ezeket végiggondolva alkottak egy nyelvet — a nyelv neve POUCHES; puszta érdekesség maradt —, mely *a típust azonosítja a mértékegységgel*, és koncepciója a következő: a felhasználó definiál néhány alapmértékegységet, ezáltal definiálva lesz ezeknek összes szorzással és osztással képzett származéka is. Definiálhat ezenkívül szinonímákat is: megmondhatja például, hogy a liter a köbméter ezredrésze. Ezután jön a konstansok és változók deklarációja; minden konstansnak vagy egyszerű, vagy származtatott típusa (mértékegysége) van. Utána sorakoznak a függvénydefiníciók, ahol megmondjuk, hogy milyen mértékegységű argumentumokból milyen mértékegységű eredményt állít elő az adott függvény.

Ezek alapján a fordítóprogram a fordítási időben minden értékadást ellenőriz. Az additív műveletek és az értékadás két operandusának mindig azonos mértékegységűnek kell lennie, a multiplikatív műveletek pedig értelemszerűen változtatják a mértékegységet.

A dolog sajnálatos szépséghibája, hogy csak skalárokra van kidolgozva, sem vektorokra, sem mátrixokra nincs, a rekordokról nem is szólva. Végig kellene gondolni továbbá a numerikus integrálást és hasonlókat, hogy ne kelljen ezeket minden típusra újra és újra definiálni.

Felmerül az a kérdés is, hogy ez a mechanizmus, ami igen jól működik a fizikai számításokban, működik-e az élet más területein is?

Típusgenerációk

Visszatérve a főcsapás irányába, a típusfogalom fejlődésében a következő lépés az Ada nyelv, amely elveit részben a PASCAL-ból, részben az algebrai típusfogalmon alapuló CLU nyelvből származtatja. Az Ada nyelv felfogása szerint egy adattípus egy értékkészletből és egy ezen végrehajtható műveletkészletből áll, amelyeket célszerű egy package-nek nevezett egységben együtt megadni. A nyelvben eleve adva van néhány beépített típus és néhány típuskonstrukciós eszköz (tömbök képzése, rekordok képzése, pointer egy adott típusra stb.), ezekből lehet a további típusokat felépíteni. Arra is lehetőség van, hogy egy létező típusból egy új, tőle különböző, de vele azonos belső felépítésű új típust származtassunk (deriváljunk); például a FLOAT típusból — amely a legtöbb programban a lebegőpontos ábrázolású adattípus — a KILÓ és a LITER típusokat deriváljuk, amelyeket ugyanúgy akarunk ábrázolni. Természetesen itt is az a kérdés, mit lehet az így konstruált új típusú adatokkal csinálni. Az Ada válasza erre a következő: az új típus nem azonos a régi típussal, sem értékadásban, sem függvényhívásban a két típus egymással fel nem cserélhető. Ezzel szemben a két típus egymásba átkonvertálható, a konverziót explicit módon megjelölve. Ezenkívül az új típus örökli az őstípus minden műveletét; pontosabban, mindazok a műveletek, amelyek az eredeti típusra adva voltak, implicit módon újradeklarálódnak a derivált típus definíciója kapcsán, de úgy, hogy ahol bennük az őstípus szerepelt argumentumként vagy eredményben, ott az új deklarációban az összes helyen az új típus fog állni. (Például a KILÓ-ra és a LITER-re adva lesz a négy alapművelet, úgy, hogy annak mindkét argumentuma és eredménye is KILÓ illetve LITER lesz. Adva lesznek a kisebb, nagyobb stb. műveletek, amelyek két KILÓ, illetve LITER mennyiséghez BOOLEAN típusú mennyiséget rendelnek stb.) Ezenfelül a felhasználónak arra is lehetősége van, hogy olyan új műveleteket (szubrutinokat és függvényeket) hozzon létre, amelyek csak az új típusra vannak értelmezve, illetve felüldefiniálhatja az őstípushoz rendelt műveletek működését.

Az algebrai típusfogalom eszméit tükrözi az Ada nyelv priváttípus fogalma. Ha egy package-ben egy deklarált típusról kijelentjük, hogy privát, akkor belső felépítése csak a package-en belül látszik. Kívülről csak annyit tudunk róla, hogy ő egy típus, amelyre az egyenlőségen és az értékadáson kívül csak azok a műveletek alkalmazhatók, amelyeket a package a külvilág számára megad. Ha pedig egy típust limitednek nyilvánítunk, arra még az értékadás és az egyenlőség sem alkalmazható, csak a felhasználó által a package-ben megadott műveletek.

Az altípus mint olyan

Az Ada nyelv újdonsága az altípus fogalma. Az altípus a típus értékkészletének részhalmaza, amelyet elnevezünk. Az altípus bevezetése lehetőséget ad rá, hogy előírjuk: akár egy változó értéke, akár egy függvény argumentuma vagy eredménye mindig az adott halmazba essék. Az Adában az altípusokra csak bizonyos jól ellenőrizhető megszorításokat tehetünk. Skalárok esetén az altípus csak egy intervallum lehet. Tömböknél, mivel a tömb típus csak a tömb elemeinek típusát és az egyes indexek típusát írja elő, az azonos alsó és felső indexhatárú tömbök alkotnak egy-egy altípust. Rekordoknál, mivel egy típuson belül több variáns is lehet, az azonos szerkezetűek alkotnak egy altípust. (Egyes Ada utáni nyelvek már ennél többféle altípus megadására is lehetőséget nyújtanak.) Az altípushoz tartozó értékek természetesen beletartoznak a típusba is. Ahová tehát egy

ejlődése II.

adott típushoz tartozó érték szükséges, oda kerülhet egy bármilyen altípushoz tartozó érték is, és viszont: ahova egy adott altípusú érték szükséges, ott elfogadunk minden értéket, amely a típushoz tartozik, de ellenőrizzük (rendszerint futásidőben), hogy a megszoritásnak eleget tesz-e.

Azt hihetnők, hogy az Ada típusfogalma már minden igényt kielégit, de ez természetesen nincs igy. A probléma ma is ott lelhedzik, ahol régebben: azaz ha bevezetünk egy új típust, amely egy korábbiból származik, akkor meg kell határozni, milyen műveletek legyenek érvényesek az új típusra. Egyrészt egyáltalán nem megnyugtató, hogy az új típus automatikusan örököl egy sor műveletet, és nincs módunk rá, hogy ezek közül bizonyosakat kihagyjunk, másrészt a vegyes típusú műveletek csak egy-egy argumentumtípusuk szerint öröklődnek: például, a FLOAT/FLOAT osztásból automatikusan létrejön a KILÓ/KILÓ és LITER/LITER, de a KILÓ/LITER típusú osztás automatikusan nem deklarálódik a FLOAT osztásból; ezt csak a felhasználó vezetheti be, de lehet, hogy ez esetben nem egyetlen gépi műveletté, hanem teljes valódi paraméterátadással járó programhívássá fordul.

Korrektebb típuselképzelés lenne, ha minden származtatott típusnál meg kellene adnunk az alkalmazható függvényeket, ahol a megadás egyik lehetséges módja, hogy közöljük: az adott függvény nem más, mint egy korábbi függvény alkalmazása új, de a régivel kompatibilis ábrázolású típusokra.

Van modern típusfogalom?

A ma uralkodó szemlélet nagy hátránya, hogy kissé túlspecifikálja a típust. Új meg új függvényeket kell bevezetni különböző típusokra, amikor az elvégzendő feladat lényegében azonos. Régen írhattunk egy általános listakezelő vagy egy általános veremkezelő programot. Az új elvek szerint ilyesmire nincs mód, mert a programot csak akkor tudjuk megírni és majd használni, ha ismerjük a verem vagy a lista elemeit.

E probléma megoldásához két út kínálkozik a modern programozási nyelvekben. Az egyik a generikus típuson, függvényen, package-en át vezet. A felhasználónak lehetősége van *egy* olyan *generikus típus* bevezetésére, amely *függ egy formális típusparamétertől*, és a típushoz definiálja a kezelőrutinokat is, melyek természetesen szintén függnek a formális típustól, és bizonyos ezt kezelő, szintén formálisan megadott függvényektől. Például a felhasználó megadhat egy generikus veremtípust, amelyben az elem típusa egy formális paraméter, megírhatja a teljes veremkezelő műveletkészletet úgy, hogy abban csupa ismert művelet szerepeljen, plusz az elemtípushoz tartozó értékadás.

Természetesen ezt a generikus típust a programunkban csak azután tudjuk használni, miután belőle konkrét példányokat készítettünk úgy, hogy a formális típust egy konkrét típussal helyettesítjük be. *A generikus típust és a hozzá tartozó függvényeket elképzelhetjük mint egy makrót, amelyből a formális paraméterekbe helyettesített típusok és függvények megadásával készítünk valódi típusokat és hozzájuk tartozó műveleteket.*

Helytelen lenne azonban a generic-et szövegmanipulációnak elképzelni; rendszerint magából a generic definíciójából látható, hogy a példányképzés nem a forrásszövegen, hanem annak egy valamennyire lefordított másán történik, így a példányonkénti újrafordítást megtakarítottuk. Ez persze mit sem változtat azon, hogy csak a forrásszöveg terjedelme csökken, a kód azonban duplikálódik a különböző példányokra.

Polymorfizmus

Ez a másik jelenleg divatos út. Egy műveletet akkor tekintünk polymorfikusnak, ha több különböző típusú argumentum állhat ugyanannak a formális paraméternek a helyén. Az irodalom *megkülönbözteti az ad hoc polymorfizmust és az igazit.* Ad hocnak nevezzük a polymorfizmust, ha tulajdonképpen két teljesen különböző függvényről van szó, csak megjelenésük hasonló (például valós számok osztása és egészek maradékos osztása). Sok programozási nyelv mindkettőt egyformán jelöli, bár a kettő eredménye csak ritkán esik egybe. Nem véletlen, hogy az utóbbit ma már az általános iskolában bennfoglalásnak nevezik. Az igazi polymorfizmus az, amikor a végrehajtandó műveletsor ugyanaz, bár maguknak az elemi műveleteknek (értékadás, egyenlőség, kisebb, nagyobb, egyszerű aritmetikai műveletek stb.) az értelmezése különböző típusok esetén lehet más és más. Interpreterrel megvalósított nyelvekben, ahol az adatokkal együtt azok típusleírása is adott, a polymorfizmus meglehetősen elterjedt, ezt azonban sokan nem tekintik igazi megoldásnak, mert a műveletek elvégezhetősége csak futás közben derül ki. Vannak viszont olyan kísérletek, mint például a Poly nyelv, amely már a fordítási idő alatt teljes típus- és elvégezhetőség-ellenőrzést folytat.

A generic és a polymorfizmus nem független fogalmak. A polymorfizmusnak egy lehetséges — bár szellemétől idegen — megvalósítása, ha a különböző típusokhoz tartozó függvényhívásokhoz különböző függvénypéldányokat rendelünk. Hasonlóképpen egy generikus függvény összes példányát implementálhajtuk egy polymorfikus függvénnyel. Pillanatnyilag úgy látszik, hogy *a generic és a polymorfizmus egy nyelvben túl sok egyszerre,* egy nyelv vagy az egyikre ad lehetőséget, vagy a másikra.

Térjünk vissza még egyszer utoljára a vegyes típusú műveletekre. Vajon miért nem lehet összeadni 6 almát és 4 körtét? Igenis lehet — és az eredmény 10 gyümölcs. Ha feltételezzük, hogy adva volt egy gyümölcs nevű típus és ennek speciális esete az alma meg a körte, akkor 4 alma+5 alma=9 alma, 3 körte+4 körte=7 körte, de a 6 alma+4 körte csak úgy értelmezhető, ha az almát és a körtét visszavezetjük egy magasabb típusba: a gyümölcsbe. A típusok között legtöbb nyelv definiál egy hierarchiát. Számos mesterségesintelligencia-kutatási projektben a fogalmak között nagyon hasonló hierarchia áll fenn, és kidolgoztak utakat arra, hogyan lehet áttérni egy általánosabb fogalomról egy speciálisabbra, valamint a speciálisról az általánosra. Ez is egy lehetséges út egy újabb típusfogalom felé.

Egyelőre *nem valószínű, hogy a közeljövőben egy új,* minden igényt kielégítő *típusfogalom születik.*

TEHÁT?

Mint a fentiekből látható, a számítástechnika eddigi rövid történelme folyamán is számos típusfogalom született, és most még csak az adat+utasítás alapon felépülő nyelveket tekintették át. Ezek a típusfogalmak egymásra épülnek olyan értelemben, hogy minden típusfogalom számos hézagot és ellentmondást hagyott. Az utánuk következő nyelvek ezeket próbálták kiküszöbölni több-kevesebb sikerrel; közben jó néhány vakvágányra futott.

Ha egy programozási nyelvnek eleme néhány alapvető adatmanipuláló lehetőség és az utasítások sorrendjét vezérlő utasítás, akkor *a nyelv matematikailag minden feladat megoldására alkalmazható. Ilyen szempontból az adattípus egyáltalán nem szükséges.* Az adattípus egy segédeszköz, amely (lehetőleg fordítási időben) ellenőrzi a program szövegét, hogy nincsenek-e benne elírások; azt, hogy az adatokkal csak a megengedett manipulációkat hajtottuk-e végre. Az ellenőrzés akkor hatékony, ha elég kemény szabályok írják elő, hogy mit szabad tenni és mit nem. *Ami tehát egyik oldalon segítség, a másik oldalon korlátozás a felhasználónak.*

Minden adattípus-fogalom mögött van egy elképzelés, hogy mi ez az adattípus és hogyan kell ezt használni. Bizonyos emberek egyes elképzeléseket nem tudnak és nem akarnak elfogadni. Ilyenkor hitviták alakulnak ki; ezekkel szakfolyóiratokban is találkozhatunk. A vitákban általában az egyik nyelvnek a másik nyelv feletti magasabbrendűségét bizonygatják, alkalmazási területtől és gyakorlati tapasztalatoktól elvonatkoztatva, puszta a hitelvek alapján.

Mindenkinek csak azt ajánlhatjuk, hogy ismerjék meg és próbálják alkalmazni a rendelkezésükre álló nyelv filozófiáját, mert akkor a program valószínűleg szebb és áttekinthetőbb lesz mind a maguk, mind mások számára. Mint láttuk, egyik nyelv sem tökéletes, és nálunk még az is ritkaság, hogy egy adott probléma megoldásához megkapjuk a célnak legjobban megfelelő számítógépet és nyelvet.

FARKAS ERNŐ

Magyar ékezetes karakterkészlet EPSON FX—100-as és 105-ös nyomtatókra

A személyi számítógépek szövegfeldolgozási, szövegszerkesztési alkalmazásainak terjedésével egyre bántóbbá válnak az ékeztek nélküli írásképek. Jó szövegszerkesztőkben szép számmal válogathatunk; ezek azonban alapvetően az angol, illetve amerikai karakterkészlettel végzett munkát támogatják. Az elterjedt nyomtatók ugyan többféle nemzeti karakterkészlettel is képesek dolgozni, de a magyar ékezetes ábécével is bíró nyomtatóval — néhány kivételtől eltekintve — nem találkozhatunk. Így vagy megelégszünk némi kompromisszum árán a például a magyarhoz még viszonylag legközelebb álló svéd karakterekkel, esetleg a német ábécével, vagy magunk látunk hozzá magyar betűkészletet csinálni FX nyomtatóinkra, esetleg néhány más, az IBM PC-khez kapcsolható nyomtatóra.

Ez utóbbi elhatározáshoz szeretnék most hozzájárulni egy igen egyszerű kis BASIC program formájában, illetve a karakterszerkesztés néhány technikai részletének ismertetésével.

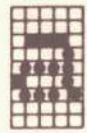
A programmal a ROM memóriában tárolt karakterkészletet (azaz a karakterek képét) a RAM-területre töltjük, a 120-as sorban felsorolt karakterek helyét pedig (ebben a karakterkészletben) a DATA sorokban definiált új karakterképekre írjuk át. A program lefuttatása után, ha a 120-as sorban szereplő karakterek előfordulnak valahol a szövegszerkesztővel írt szövegben, vagy akár egy BASIC program szövegében, a nyomtatón mint az általunk definiált ékezetes karakterek jelennek meg. Így hát magunk határozhatjuk meg, hogy mely billentyűk helyére telepítjük új betűinket.

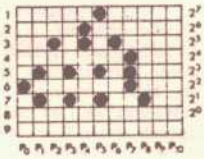
A programot a lehetséges megoldások egyikére tett javaslatnak is tekinthetjük.

Nézzük ezután a betűszerkesztés technikáját. A személyi számítógépekhez használatos nyomtatók többsége pontmátrix nyomtató: a karakterek egy képzeletbeli rácsban jelennek meg. A képük attól függ, hogy ennek a rácsnak mely vonalai kereszteződésénél vagy mely kockáiba tesz a nyomtató egy pontot. Az „a" betű képe például az *1. ábrán* látható.

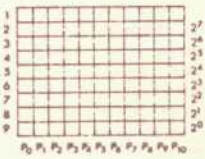
A rácsot a betű tartójának hívjuk. A tartó (angolul: font) magassága 9 pont, szélessége látszólag 5 pont. Előfordul azonban, mint az 1. ábrán is, hogy egy nyomtatott pontnak a betűkép érdekében éppen a vonalra kell esnie. Ezért a betűt egy 9×11 pontos tartóban tervezzük meg, de ezt a rácsot sem használjuk ki teljesen: vagy a legfelső, vagy a legalsó pontsora használatlan marad. Így a tervezésnél mindig csak 8 sorral számolhatunk.

1. ábra



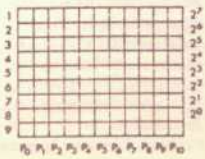


2. ábra



3. ábra

4. ábra



```
10 DIM B$(12)
20 LPRINT CHR$(27);"6";
30 LPRINT CHR$(27);":";CHR$(0);CHR$(0);CHR$(0);
40 LPRINT CHR$(27);"%";CHR$(1);CHR$(0);
50 FOR I=1 TO 12:READ B$(I):NEXT I
60 FOR I=1 TO 12
70 READ A,B,C,D,E,F,G,H,J,K,L
80 LPRINT CHR$(27);"&";CHR$(0);B$(I);CHR$(139);
90 LPRINT CHR$(A);CHR$(B);CHR$(C);CHR$(D);CHR$(E);
100 LPRINT CHR$(F);CHR$(G);CHR$(H);CHR$(J);CHR$(K);CHR$(L)
110 NEXT I
120 DATA '',"",qq,QQ,[[,((,)),]],@@,&&,\\,^^
130 DATA 4,10,32,10,96,138,32,28,2,0,0:REM á betű
140 DATA 6,8,20,32,68,160,20,8,6,0,0  :REM A betű
150 DATA 28,34,8,34,72,162,8,34,24,0,0:REM é betű
160 DATA 62,0,42,0,106,128,42,0,34,0,0:REM É betű
170 DATA 0,28,162,0,34,0,162,28,0,0,0 :REM ó betű
180 DATA 28,162,0,34,0,34,0,162,28,0,0:REM o betű
190 DATA 0,60,128,2,0,2,128,60,2,0,0  :REM ú betű
200 DATA 60,130,0,2,0,2,0,130,60,0,0  :REM U betű
210 DATA 0,28,34,0,98,128,34,28,0,0,0 :REM ő betű
220 DATA 0,60,0,2,64,130,0,60,2,0,0   :REM ű betű
230 DATA 0,28,34,64,162,0,98,156,0,0,0:REM ő betű
240 DATA 0,60,0,66,128,66,128,60,2,0,0:REM ű betű
```

Helyezzük el például egy „a" betű képét egy ilyen tartóban *(2. ábra)*. Tekintsük úgy ennek a tartónak minden oszlopát, hogy megfeleltetünk nekik egy bájtot. A legalsó pont a 0. helyiértékű bitet reprezentálja, a legfelső a kettő 7. hatványának felel meg. Olvassuk le az egyes bájtokban (oszlopokban) található értékeket úgy, hogy azok kettes számrendszerbeli számok, és a pontok helyére 1-eseket képzelünk. Így tehát az első oszlop bájtértéke például 4, a másodiké 10, a harmadiké 32 stb. Látható, hogy ahol az eredeti tartóban egy pont vonalra esett, azt ebben a „kiterített" formában úgy ábrázoltuk, hogy egy páratlan sorszámú oszlopba helyeztük a neki megfelelő pontot. Az egyes bájtok leolvasott értékei kerülnek a DATA sorokba: egy-egy sorba az egyes karaktereket leíró bájtsorozat.

Ezután meg kell határoznunk a nyomtatást vezérlő bájt értékét. Ez dönti el, hogy a betű hogyan helyezkedik el a tartóban. Ha a 7 helyiértékű bit 0, akkor a kilencedik (azaz a legalsó) pontsor is ki van használva *(3. ábra)*; ha értéke 1, akkor nincs *(4. ábra)*.

A 4, 5, 6 helyiértékű bitek a karakter kezdőoszlopát, a 0, 1, 2, 3 helyiértékűek pedig a karakter végét határozzák meg a tartóban. A mi esetünkben ez az érték minden karakterre egységesen 139.

Végül a program fontosabb sorainak tartalma:

20 Lehetővé teszi a 128 és 159 közé eső ASCII kódú karakterek nyomtatását. Erre akkor lehet szükség, ha 128-nál nagyobb ASCII kódú karakter helyére akarunk új karaktert beírni.

30 Bemásolja a ROM karakterkészletet a felhasználó által definiált karakterkészletbe.

40 Átkapcsol az új karakterkészlet használatára. Ezentúl a RAM-ból veszi a nyomtató a karaktermintákat.

80 Lehetővé teszi a karakterdefiníciót. A B$(I)-ben aktuálisan levő betű képe alakul át. A sor végén a vezérlőbájt értéke szerepel.

90—100 Átírják a karakterek képét.

Természetesen a leírt módon más karakterek is definiálhatók, tetszőleges billentyűkre telepíthetők. Ez utóbbi a 120-as sor átírásával valósítható meg.

Az FX—100-as, FX—80-as, FX—105-ös, FX—85-ös nyomtatókon a DIP 1 kapcsoló 6., 7. és 8. pozíciójának (a két utóbbi nyomtatón a 4. pozíciónak is) ON állásban kell lennie.

DR. JÁNOSA ANDRÁS

μINFORM

A tartalomleírások az alábbi folyóiratokban megjelent programlistákról készültek:

A folyóirat neve	Kódja
64'er Magazin	64er
Chip Magazin	chip
Commodore Horizons	coho
Commodore Microcomputers	comi
Compute!	cute
Computer Persönlich	pers
Happy Computer	happ
hc - Mein Home-Computer	hc
mc - Zeitschrift	mc
Run /USA/	run
Sinclair User	sinc
Your Sinclair	ysin

A tartalomleíró szóláncokat permutáltuk, a szólánc variánsokat pedig alfabetikusan rendeztük.

A tartalomleírás egy szóláncból áll, majd a listában ezt követi a forrás megjelölése a folyóirat azonosítójával, a megjelenés dátumával és a cikk előkeresésához a kezdő oldalszám és a terjedelem megadásával. A mellékelt lista értelmezéséhez még az alábbiakat kell tudni. A tartalomleírás szóláncában elsőként a téma átfogó megnevezése, utána a számítógéptípus(ok), ezt követően a szűkebben jelölt tartalom meghatározása szerepel, majd esetlegesen néhány, a közleményt minősítő adat (például : cikksorozat).

A forráshely karaktersorozatát nyíl vezeti be, melyet a / jelig a folyóiratok azonosítója, a két / jel között az évszám, folyóiratszám és kötőjellel a kezdő oldalszám követi, a végén pedig a közlemény teljes oldalterjedelme áll.

A folyóiratok a SZÁMALK szakkönyvtárában (Budapest XI., Szakasits Á. út 68. Nyitva: 8-tól fél 5-ig. Tel.: 853-111/251) is follelhetők. A kiválasztott anyagról másolat rendelhető az alábbi formában:

SZÁMALK Szakkönyvtára
Budapest, 112. Pf.: 146. 1502

Megrendelem a Mikroszámítógép Magazin 1987/ sz. alapján a következő folyóirat-oldal-másolatokat:

Kód: _____ Példányszám: _____
Kód: _____ Példányszám: _____
Kód: _____ Példányszám: _____

A megrendeléshez csatolom az oldalankénti 8,- Ft-os szolgáltatási díj befizetését igazoló csekkszelvényt.
Dátum, név, pontos cím.

PROGRAM.LISTA
sdatatvitel/cikksorozat/commodore 64 az rs232 kezelese
->run2/86.06-30/3
PROGRAM.LISTA
assembler programozas/cikksorozat/commodore 64/jatekprogram keszites
->happ/86.07-54/2
PROGRAM.LISTA
atari 520 st/disassembler
->hc/86.07-37/6
PROGRAM.LISTA
atari xl/xe/hasznos segedprogramok
->hc/86.07-43/3
PROGRAM.LISTA
basic programozas/commodore 64/(mikro-compiler) a programfutas kb. tizszeres gyorsitasahoz
->run2/86.07-83/5
PROGRAM.LISTA
commodore 128/botkormany teszteles
->run2/86.06-50/2
PROGRAM.LISTA
commodore 128/grafika/(ultra hires)/3d oszlopdiagramok/14 uj utasitas
->run2/86.06-70/11
PROGRAM.LISTA
commodore 128/grafikus tartartalom kiolvasasa-atalakitasa-nyomtatasa
->hc/86.07-71/2
PROGRAM.LISTA
commodore 128/jatekprogram/(vectors)/akciojatek ->64er/86.07-73/4
PROGRAM.LISTA
commodore 128/kepernyokezeles/80-oszlopos modus c64 programokhoz
->run/86.07-46/3
PROGRAM.LISTA
commodore 16/grafika/gepi rutinok
->run2/86.07-42/2
PROGRAM.LISTA
commodore 64/(full-screen editor)/basic programsor-szoveg atalakitas
->chip/86.07-114/4
PROGRAM.LISTA
commodore 64/(needlegraph)/himzester v felepitese szines mozaikokbol
->run/86.07-34/4
PROGRAM.LISTA
commodore 64/folyamatabra szerkeszte s din 66001 szimbolumokkal
->run2/86.07-52/9
PROGRAM.LISTA
commodore 64/fuzerbevitel elozetes karakterszam-megadassal
->hc/86.07-72/2
PROGRAM.LISTA
commodore 64/grafika/(hi-res writer) szovegiras a monitorkepre
->run/86.07-50/3
PROGRAM.LISTA
commodore 64/grafika/cimkep szerkeszte s/tarolas/12 teteles menu
->run2/86.07-66/9
PROGRAM.LISTA
commodore 64/grafika/muveszi grafika szines kepek/kepszegelyek eloallitasa egyszeru fuggvenyekkel
->hc/86.07-74/3
PROGRAM.LISTA
commodore 64/jatekprogram/(the pink panther)/labirintusjatek
->hc/86.07-77/7
PROGRAM.LISTA
commodore 64/kiegeszites a (86.04) szamban kozolt hypra-basic listahoz
->64er/86.07-96/2
PROGRAM.LISTA
commodore 64/lemezegyseg(1541)/(ultraboot menue)/file-pool gyorstolto
->happ/86.07-67/4
PROGRAM.LISTA
commodore 64/neuzroom/ekezetes betuk kel valo kiegeszites
->64er/86.07-89/2
PROGRAM.LISTA
commodore 64/nyomtato(epson/mps-802/rs232)/(master-text) kiegeszites/jelkeszlet generalas ->64er/86.07-67/5
PROGRAM.LISTA
commodore 64/programbetoltes katalogol-hazas atalakitasa-nyomtatasra
->hc/86.07-71/2
PROGRAM.LISTA
commodore 128/jatekprogram/(vectors)/akciojatek ->64er/86.07-73/4
PROGRAM.LISTA
commodore 128/kepernyokezeles/80-oszlopos modus c64 programokhoz
->run/86.07-46/3
PROGRAM.LISTA
commodore 16/grafika/gepi rutinok
->run2/86.07-42/2
PROGRAM.LISTA
commodore 64/(full-screen editor)/basic programsor-szoveg atalakitas
->chip/86.07-114/4
PROGRAM.LISTA
commodore 64/(needlegraph)/himzester v felepitese szines mozaikokbol
->run/86.07-34/4
PROGRAM.LISTA
commodore 64/folyamatabra szerkeszte s din 66001 szimbolumokkal
->run2/86.07-52/9
PROGRAM.LISTA
commodore 64/fuzerbevitel elozetes karakterszam-megadassal
->hc/86.07-72/2
PROGRAM.LISTA
commodore 64/grafika/(hi-res writer) szovegiras a monitorkepre
->run/86.07-50/3
PROGRAM.LISTA
commodore 64/grafika/cimkep szerkeszte s/tarolas/12 teteles menu
->run2/86.07-66/9
PROGRAM.LISTA
commodore 64/grafika/muveszi grafika szines kepek/kepszegelyek eloallitasa egyszeru fuggvenyekkel
->hc/86.07-74/3
PROGRAM.LISTA
commodore 64/jatekprogram/(the pink panther)/labirintusjatek
->hc/86.07-77/7
PROGRAM.LISTA
commodore 64/kiegeszites a (86.04) szamban kozolt hypra-basic listahoz
->64er/86.07-96/2
PROGRAM.LISTA
commodore 64/lemezegyseg(1541)/(ultraboot menue)/file-pool gyorstolto
->happ/86.07-67/4
PROGRAM.LISTA
commodore 64/neuzroom/ekezetes betuk kel valo kiegeszites
->64er/86.07-89/2
PROGRAM.LISTA
commodore 64/nyomtato(epson/mps-802/rs232)/(master-text) kiegeszites/jelkeszlet generalas ->64er/86.07-67/5
PROGRAM.LISTA
commodore 64/programbetoltes katalogus-menu alapjan ->chip/86.07-117/2
PROGRAM.LISTA
grafika/kepgeneralas matematikai fuggvenyekkel/peldak ->happ/86.07-56/2
PROGRAM.LISTA
jatekprogram/pascal nyelv
->hc/86.07-63/3
PROGRAM.LISTA
meresztechnika/kozepertek es kozepes hiba szamitas ->hc/86.07-73/2
PROGRAM.LISTA
pascal nyelv/szovegvisszamentes cp/m 80-turbo 3.0 alatt
->hc/86.07-66/1
PROGRAM.LISTA
run2/listabegepelesi segedletek
->run2/86.06-36/5
PROGRAM.LISTA
sinclair ql/sokteteles katalogus tobbhasabos megjelenitese
->chip/86.07-130/1
PROGRAM.LISTA
sinclair spectrum/(input #2)/megjelenites a kepernyo felso szektoraban
->hc/86.07-49/2

Integrált szoftver

Mielőtt rátérnénk a POSTÁZÓ (például címkekészítő) program ismertetésére, néhány további információval segítjük azokat, akik a szövegszerkesztő programot használni akarják.

A nyomtatóvezérlő ASCII kódok

Ezeket rendszerint a nyomtatás elején, együtt szokás kiküldeni a nyomtatóra. Ha menet közben is váltani szeretnénk, például dőlt betűre, az ASCII vezérlőkódnak meg kell előznie a szöveget. Ezek a karakterek általában nem nyomtathatók, de a nyomtatandó szöveg részének tekinti őket a program. Ezért a szöveg annyiszor balra tolódik a képernyőn levő írásképhez képest, ahány ASCII vezérlőkódot adtunk ki. Tabulációval állíthatjuk helyre az eredeti formát.

Célszerű gyakran készíteni másolatot a begépelt szövegről. Így nem túl nagy probléma, ha esetleg a program valamely hibás utasítás miatt „elszáll", vagy hálózati zavar miatt elvész az addig begépelt szöveg. A biztonság kedvéért használjunk munkakazettát, és a már kész szövegeket külön kazettán tároljuk.

A BREAK gomb lenyomására a program leáll. A képernyőn a BASIC-ben szokásos hibaüzenet jelenik meg. Ha a BREAK gombot csak azért nyomtuk meg, hogy a munkát kis időre felfüggesszük, akkor CONT, majd <ENTER> beírása után a program újra indul. Ha főmenübe akarunk visszatérni, írjuk be: GOTO500 <ENTER>. Ha nem történt módosítás vagy listázás, a begépelt szöveg érintetlen marad.

Grafikus karakterek használata

Minden grafikus karakter (ilyennek tekinthetők a magyar ékezetes betűk) 5 helyet foglal egy sorban, így a maximum 50 lesz. A soremelés minden ASCII sor végén automatikus. A formátum: /aNNN. Így a grafikus karakterek a szövegben együtt használhatók a többivel.

Szöveg beadása a billentyűzetről nyomtatás közben

Az utasítás: /k, és egy képernyőre kerülő üzenet követheti. Amikor a program a /k utasításhoz ér, a nyomtatás leáll, az üzenet megjelenik a képernyőn. Amit ezután begépelünk, nyomtatáskor az „üzenet" helyére kerül. Például:

/k A MAI DAATUM
/s2/k NEEV
/k CIM ELSOE SORA
/k CIM MAASODIK SORA

Megjegyzendő, hogy a /k-t a program utasításként kezeli, és ezért mindig megszakítást eredményez.

Jó tudni, hogy ha elmarad az „új bekezdés" utasítás törlése, akkor ez lesz a kezdő érték a következő nyomtatásnál is, függetlenül attól, hogy szöveget vittünk ki a szalagra, és onnan például egy másikat töltöttünk be.

Szövegrészlet nyomtatása

Tételezzük fel, hogy egy tízoldalas szöveg 3. oldalát akarjuk csak újra kinyomtatni, mert ott néhány fontos táblázat van. Erre két lehetőség adódik:

— Indítsuk a nyomtatást a 3. oldalon, és nyomjuk le az <U> gombot, amikor a 4. lap kezdődik.

— Vigyünk be egy üres sort a 3. lap után, és hagyjuk, hogy a nyomtató magától leálljon.

Az üres sort valamely szövegpuffer nem használt részéről kell a kívánt helyre „mozgatni", és nem szabad „beszúrni". Az üres sor két blanket tartalmaz, ami leállítja a nyomtatót. Ne felejtsük el, hogy ezt az üres sort el kell távolítani, amikor a nyomtatással végeztünk; a szöveg így ismét helyesen lesz a szalagon.

Hosszú szövegrészletek mozgatása

Ha a gépben elég szabad memória van, ez a művelet nem vesz túl sok időt igénybe. Ahogy nő a szöveg által elfoglalt memóriaterület, úgy növekszik a mozgatáshoz szükséges idő is. A mozgatás ennél a megoldásnál soronként halad. A módszer nem túl gyors, de kevés szabad memóriát igényel, ezért a kisebb gépeken is elboldogulunk vele.

Még egy megjegyzés. Néha, egy-egy mondat bevitele után úgy tűnik, mintha a gép „lefagyott" volna. A program működése közben sok átmeneti szövegsztringet és mutatót készít, amelyekre a felhasználás után már nincs többé szükség. Futtatáskor ezekkel a „maradékokkal" nem törődik addig, amíg új munkaterületeket talál a memóriában. Ha már nincs, a program futása felfüggesztődik, és kezdetét veszi a „nagytakarítás". A felszabadított munkaterületen folytatódik a szerkesztés.

POSTÁZÓ

Most nézzük az előző számunkban közölt POSTÁZÓ program használatát. Szokás szerint töltsük be kazettáról a már előzőleg begépelt programot. CLOAD "POSTAAZO" <ENTER>, majd indítsuk el: RUN. A program megkérdezi, hogy új fájllal dolgozunk-e, vagy nem. Ha nem, akkor helyezzük be a programkazetta helyére a megfelelő adatkazettát; most betölti a gép a használni kívánt „régi" fájlt.

Gépeljük be a dátumot. Ezek után a számítógép közli velünk a legutóbbi adatfrissítés dátumát. Ekkor kell kiválasztani a képernyőn megjelenő instrukció alapján a nyomtató típusát is (soros/párhuzamos). Ha a program „eltéved", viszatérhetünk a főmenühöz a GOTO1000 utasítással.

A legfontosabb választási lehetőségek

U = fájl frissítése
P = egy fájl nyomtatása
L = címke nyomtatása
S = sorba rendezés
T = nagybetű/kisbetű használata
E = kivonat készítése az adat-összefésülőprogram számára
Q = fájl letárolása szalagra és visszatérés a főmenühöz

Ha kilépünk a programból (<Q>), az új címfájl automatikusan eltárolódik.

Ahhoz, hogy az egyes almenüket megvizsgáljuk, célszerű néhány adatsort felvenni. Így könnyebb lesz a gyakorlás.

A fájl frissítése (<U>) a következő választási lehetőségeket adja:

A = új rekord hozzáírása a fájlhoz
D = a rekord törlése
C = egy rekord cseréje
R = visszatérés a főmenühöz

<Q> lenyomásával szintén visszatérhetünk a főmenübe, de nem szabad elfelejteni, hogy a Q gomb kétszeri lenyomásával visszatérhetünk a BASIC interpreterhez, és a program a fájl frissítését befejezettnek tekinti.

Az <A> gomb lenyomása után a képernyőn a következő ábra jelenik meg:

POSTAAZO

1:
2:
3:
4:
5:
6:
7:

8: /KOD MEZOE/ =

Megszolitas /Dr, PROF, MR, Mrs stb./

> ?

Általában a következő sorrendben kell kitölteni a képernyőt (bár a program a legtöbb mezőnél nem limitálja a karakterszámot, célszerű mértéket tartani):

1. megszólítás
2. családi név (az esetleges kiegészítésekkel: K. Szabó vagy Nagy II. stb.)
3. utónév (a foglalkozás megjelölésével — például igazgató et. — együtt)
4. a címzés első sora (az intézmény neve stb.)
5. a címzés második sora (például irányítószám, városnév)
6. a címzés harmadik sora (utca, házszám, emelet stb.)
7. esetleges telefonszám, telexszám vagy országnév
8. 10 karakternyi hely, amelyre speciális, saját tervezésű kód kerülhet

Egy rekord cseréje

A rekord kereshető a családi név, a kódmező stb. szerint. A név szerinti keresést akkor alkalmazzuk, ha mondjuk címváltozást vagy telefonszám-változást szeretnénk átvezetni. Hosszabb neveknél célszerű a keresést 4 vagy 5 karakterrel kezdeni. Kevesebb betű használata gyorsabb keresést eredményez.

Egy rekord törlése

A rekordcseréhez hasonlóan indítható. Amikor a program megtalálta azt az első rekordot, amely a keresés kulcsának megfelel, megkérdezi: „Ez az? (I/N)". Igen válaszra a rekord törlődik, nemleges válasznál folytatódik a keresés.

Visszatérés a főmenübe

Az <R> gomb lenyomásának hatására a főmenü jelenik meg a képernyőn. Hasonló eredményt érhetünk el <Q>-val is.

A kódmező használata

Ez az a pont, ahol a felhasználó a saját ötletei szerint kialakított kóddal valóban hatékonnyá teheti a keresőrutint. Ha a kódolás jól sikerült, hasznos szolgálatokat tehet; ha nem, akkor sűrűn kell cserélgetni. A kódmező szabad formátumú, csak a hossza rögzített: max. 10 karakter. Egy egyszerű ötlet a következő:

kódolandó: Bankett június 10-én, diákok és tanárok részvételével

A kódmező ekkor például így néz ki:

B:06/10<D vagy B:06/10<T

Ez az egyszerű kód is többféle módon használható.

B:06/ azt jelenti például, hogy listát kérünk az összes bankettről, ami júniusban esedékes;

06/ mindenkiről listát kérünk, akivel júniusban van megbeszélésünk;

<D listát kérünk minden diákról vagy

<T tanárról, függetlenül a dátumtól stb.

A lehetőségek, kombinációk száma szinte végtelen. A kód szerkezetét először tervezzük meg és szerkesszük meg papíron. Ne legyen túl bonyolult, és igyekezzünk könnyen megjegyezhető, logikus kódokat alkalmazni.

Jó tudni, hogy a kereséskor nem lényeges az általunk megadott kulcs pozíciója a kódon belül; a program meg fogja találni. Viszont ügyeljünk a kódszón belüli karaktersorrendre! Egyszerre csak egy kód alapján kereshetünk, tehát nem lehet utólag relációkat felépíteni a kódok között.

Ha nyomtatási programrészből hívjuk a kereső rutint, a program megkérdezi: „Minden rekordot nyomtatni kell?" Ha nem, akkor további „szűkítő keresés" következhet. A keresési kulcs egy karaktersztring lesz.

A program gyakran megkérdezi: „Minden rendben? (I/N)". Ha válaszunk igen, akkor végrehajtódik a kijelölt műveletsor, ha nem, akkor lehet elölről kezdeni például a címke kitöltését.

Fájl nyomtatásakor a név- és címinformáció balra zárt formában jelenik meg a nyomtatási képen. Címke nyomtatásánál jó segítség a POSTÁZÓ program nyomtatóbeállító rutinja. A <P> gomb lenyomására indul a kért művelet. A kódmező nyomtatása választható.

Fájlok sorba rendezése

Az új rekordokat mindig a fájl végére írjuk. Ezért, amint az új adatok begépelésével elkészültünk, el kell végezni a rendezést. Ugyanígy kell eljárnunk, ha a kulcsként használt mezőkön adatfrissítés volt.

A nagybetű/kisbetű konverzió (<T>) jól szolgál akkor, ha a neveket a nyomtatásban csupa nagybetűvel szeretnénk látni.

Ha a munkánkat befejeztük, tároljuk szalagra az elkészült fájlt. Ez automatikusan bekövetkezik <Q> lenyomására. Az adatkazetta és a programkazetta feltűnően különbözzön egymástól, esetleg még színre is. Az adatfájlból is mindig legyen másolatunk, lehetőleg nem ugyanazon a szalagon, amelyről az adatsort beolvastuk.

GALINA FERENC

ADOK—VESZEK —CSERÉLEK

Ebben a rovatban rövid, szöveges, a mikroszámítógépekkel kapcsolatos hirdetéseket közlünk. A díjszabás: közületeknek gépelt soronként (60 karakter) 100,- Ft, magánszemélyeknek az első sor 50,- Ft, minden további sor 20,- Ft. Az NJSZT tagjainak az első három sor ingyenes. Hirdetéseiket a szerkesztőség címére várjuk.

ADOK

C64-hez bővítő portra csatlakoztatható reset, és super-reset gomb 220 Ft-ért eladó (utánvéttel elküldöm). Cím: Tárkány László, Szeged, Rózsa Ferenc sgt. 128/a. 6726.

Commodore VC20-as programok eladása: Különböző felhasználói, játék és egyéb gépi kódú programok. Kérjen tájékoztatót! Levélcím: Juhász György, Salgótarján, Pf.: 157. 3100.

Programok C/Plus 4 bővített C16 számítógépekre: FORTH nyelv /nem tiny-forth!/ 450 szó, 5 screen, diszk, printer és kiemelt magnókezelő utasítások. C/Plus 4 és C64 összehasonlító rutin- és rendszerváltozó táblázat. Turbo programok /C64 Turbo-tape kompatíbilis változat is!/. Turbo file: kazettás fájlkezelést tízszerre gyorsító program, BASIC bővítéssel. SYSTEM C/Plus 4 11 kbájtos rendszerprogram BASIC és assembler fejlesztéshez. HEADER ADJUST fejbeállító program. Disk-monitor. Turbocopy diszkhez és magnóhoz. C64-es program: MONITOR PLUS /a háttér RAM-okat is kezeli/. A programok ára: 180-600 Ft, kazettán, utánvéttel, részletes kezelési utasítással. Kérésre ismertetőt küldök. Cím: Pelsőczi Gyula, Szilasliget, Ady Endre út 36. 2145.

Primo (64 kbájt) billentyűs számítógép hozzávaló tádjos magnóval, húsz műsoros programmal, egy bemutató kazettával és átjátszó kábellel eladó. Érdeklődni telefonon vagy levélben: Harangozó Péter, Budapest, Hernád u. 54. 1078. Tel: 212-852 (16 órától).

Spectrum (48 K) + Turbo Interface + 2 db Joystick + Phonemark datasette magyar nyelvű kézikönyvekkel, 25 kazettányi programmal eladó. Ára: 25 000 Ft. Érdeklődni telefonon 426-274 15 óra után.

ZX-Spectrum (48 k) számítógép, ötvenharminc programmal és leírásokkal eladó. Drahovszky Zoltán, Budapest, Ujlak u. 128. III/10. 1173.

ZX-Spectrum (48 k) kempston interfész resettel, joystickkel, felhasználói és játékprogramokkal /TOMAHAWK, COMMANDO, TAPPER/ eladó. Kerekes László, Leninváros, Kilián-köz 8. 3580.

ZX81 (64 k) bővítővel, szakkönyvekkel, játékprogramokkal 9000 Ft-ért eladó. Cím: Mikó András, Budapest, Szedres u. 10. 1213. délután.

VESZEK

C16-os, C/Plus 4-es játékprogramokat veszek és cserélek. A programlistákat a következő címre várom: Németh Tamás, Dunaújváros, Alkotás út 10. II/1. 2406.

MZ 821 Sharp 800 számítógéphez magyar nyelvű leírást és játékprogramokat vennék. Varga Sándor, Biharkeresztes, Felszabadulás u. 134/b. 4110.

CSERÉLEK

ATARI 800 XL tulajdonosok ismeretségét keresem, tapasztalatcsere céljából. Gergely Miklós, Keszthely, Zalka Máté u. 1. 3/15. 8360.

C16, C/Plus 4-es programokat cserélnék. Surányi Péter, Miskolc, Kőrösi u. 7. 3527.

Commodore 1702-es monitort cserélek 1901-re. Einhorn Róbert, Budapest, Bartók Béla u. 18. 1191.

MSX rendszerű számítógéphez programokat cserélek. Címem: Jancsurák István, Miskolc, Dráva u. 7. 3528. Telefon: /06-46/ 12-634.

Primóra írt programokat cserélek: Somogyi György, Budapest, Dob u. 94. 1073.

Egyszerű, Z80 alapú mikroszámítógép

Bizonyára sok-sok, a számítástechnikát kedvelő — de inkább csak programozással foglalkozó — felhasználó szeretne olyan berendezést készíteni, amelyik az általa alkotott programmal működik. Ez az álom nem is túl drágán valósággá válhat, ha a következőkben ismertetett mikroszámítógépet elkészítjük, és program írásával „lelket lehelünk bele".

Természetesen így, szavakban ez nagyon könnyű, de a megvalósításhoz azért dolgozni is kell. Lássunk hozzá!

Magát az áramkört az *1. ábra* mutatja, és a „Világ egyik legegyszerűbb Z80 alapú mikroszámítógépe" nevet viseli. (Azért nem a legegyszerűbb, mert az eredetit, az ötletet adó áramkört a Radio-Electronics c. folyóirat valamelyik 1986-os számában láttam.) A rajzon megtalálhatók mindazok a részegységek, amelyek egy működő mikroszámítógéphez szükségesek:

— a központi egység (CPU): ez a Z80;

— a memória, amelyben tárolt programot a központi egység végrehajtja: ez a 2716 típusú EPROM;

— a külvilággal való kapcsolatra a be- és kimenetek: a 74LS244 típusú, 8 bites bemeneti és a 74LS273 típusú 8 bites kimeneti áramkör;

— a központi egység működéséhez szükséges órajelhez és a vezérlőjelek létrehozásához a fennmaradó további két áramkör.

A sarki elektronikai alkatrészboltban (amely lehet, hogy néhányunknak egy kissé messze van . . .) a bevásárlást segítendő, íme a részletes anyagjegyzék.

Áramkörök

1 db Z80 központi egység	(V1)
1 db 2716 EPROM	(V2)
1 db 74LS244 IC	(V3)
1 db 74LS273 IC	(V4)
1 db CD4049B IC	(V5)
1 db 74C32 IC	(V6)

Ellenállások

2 db 330 ohm/125 W	(R6, R3)
1 db 47 kohm/125 W	(R2)
10 db 4,7 kohm/125 W	(a többi)

Kondenzátorok

1 db 100 pF	(C1)
1 db 10 mikroF/6 V	(C2)

Egyéb

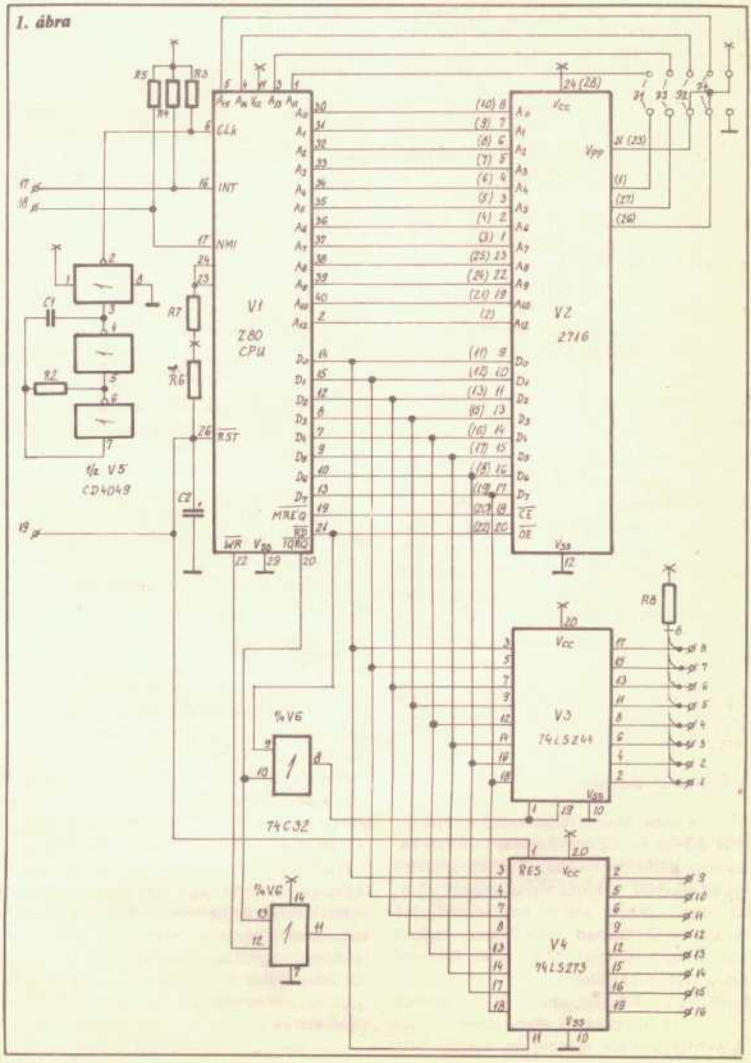
1 db áramköri lap
1 db 24 lábú IC foglalat
a többi foglalat

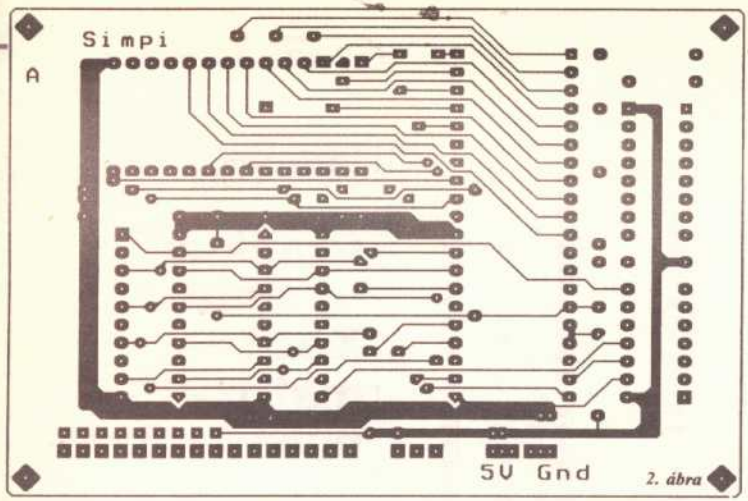
A figyelmes olvasók bizonyára észreveszik, hogy a felsorolásban nincs írható-olvasható memória RAM, ami pedig feltétlenül szükséges a programbeli adatok, változók tárolásához. Ez a hiány azonban csak látszólagos, hiszen itt vannak a Z80 regiszterei, amelyek ezekre a célokra felhasználhatók.

A processzort működtető órajelet a CD4049 IC-ben lévő inverterekből kialakított RC oszcillátor állítja elő, amelynek frekvenciája közelítőleg 1 MHz. A be/kimeneti áramkörök vezérlőjeleit a 74C32 típusú, négy VAGY kapuból álló áramkör adja.

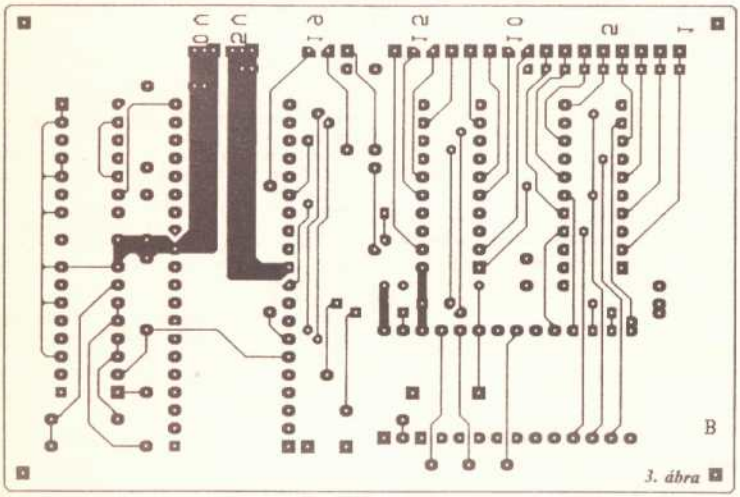
Nézzük meg, milyen korlátozásokat kell betartanunk, illetve mely lehetőségekkel élhetünk az EPROM-ba kerülő működtető program írásakor!

1. ábra





2. ábra



3. ábra

— Mivel rendszerünk külön RAM memóriát nem tartalmaz, nem használhatunk olyan utasításokat — és nem oldhatjuk meg úgy a programszervezést —, hogy a vermet használjuk, mivel ezt a Z80 processzor a memóriában helyezi el. A PUSH, POP, CALL, RET utasításokról van szó.

— A be-, illetve kimeneti utasítások portcímei tetszőlegesek lehetnek, mivel bármelyik OUT utasítás az akkumulátor tartalmát a 74LS273 kapu kimeneteire írja, és bármelyik IN utasítás a 74LS244 bemeneteire adott bináris kódot olvassa be az akkumulátorba.

— Az EPROM-ban lévő programot három programcímről is elindíthatjuk, ami azt jelenti, hogy három különböző programot is futtathatunk, az indítási címtől függően. Ezek a címek a következők.

0000H cím. RESET után vagy tápfeszültség bekapcsolásakor erről a címről kezdődik a program végrehajtása.

0038H cím. A processzor INT lábára adott nulla szintű jel hatására a programszámlálóba ez a cím töltődik, és a programvégrehajtás innen indul.

0066H cím. A processzor NMI lábára adott nulla szintű jelre a programszámlálóba ez a cím töltődik, és a programvégrehajtás innen indul.

Mivel mondandónk szándéka az is, hogy eloszlassuk a mikroprocesszoros rendszerek hardverjéhez kapcsolódóan a programozók „félelmét", vizsgáljuk meg részletesen, hogyan tudjuk a leírt áramkör működőképességét ellenőrizni. Ehhez feltételezzük, hogy rendelkezésre állnak a fent felsorolt alkatrészek, továbbá a nyomtatott áramköri lap (a NYÁK), valamint egy 5 V-os tápegység és egy oszcilloszkóp.

Az építés megkönnyítésére a *2.*, *3.* és *4. ábrán* közöljük a kapcsolás nyomtatott áramköri tervét és az alkatrészek beültetési rajzát. Csak érdekességképpen megemlítjük, hogy a tervezésbeli eredményt számítógép közreműködésével kaptuk és nyomtatón nyomtattuk ki. A kivitelezés első lépéseként szemrevételezéssel ellenőrizzük, hogy nincs-e a NYÁK-on látható zárlat vagy szakadás; a munkát csak ezek elhárítása után folytassuk!

Ültessük be — helyes pozícióban! — a CD4049 áramkört és a rezgést létrehozó RC elemeket (R, C1)! Kapcsoljunk 5 V-os tápfeszültséget a kártyára, majd ellenőrizzük az áramfelvételt: legfeljebb néhány mA lehet. Oszcilloszkóppal ellenőrizzük a rezgést (az IC 2. lába). Ennek meg kell jelennie a központi egység 6. lábán is.

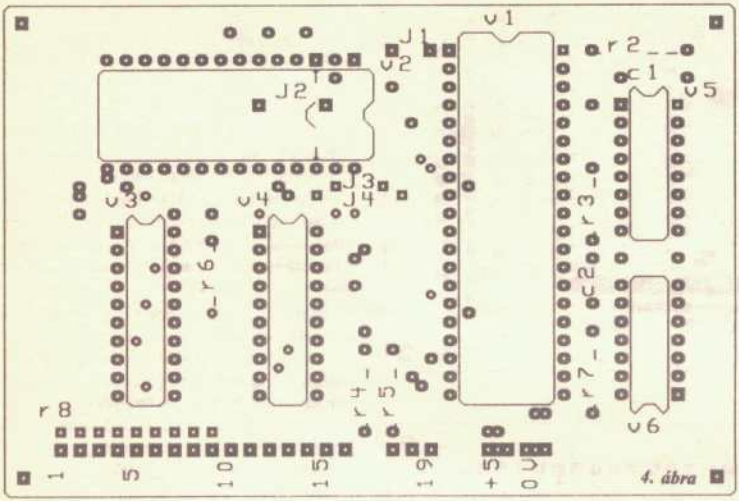
Következő lépésként ültessük be a Z80-at (pozíció!) vagy foglalattal vagy anélkül, közvetlenül. Ismét ellenőrizzük az áramfelvételt (nem lehet több, mint 200—300 mA), majd forrasszuk be a többi áramköri elemet is, a be- és kimeneti áramkörök kivételével. A RESET pont és a föld közé tegyünk egy nyomógombot. Ha megnyomjuk, RESET jel generálódik.

A Z80 adatlapjáról is leolvasható, hogy a RESET jel aktív állapotában (Ø V) a vezérlőjelek (RD, WR, IORQ, MREQ, M1) inaktív (magas) állapotba kerülnek — ezt ellenőrizzük a RESET gomb megnyomásával.

Ha megvagyunk, megpróbálkozhatunk a rendszer elindításával, azaz megnézhetjük, hogy képes-e a központi egység végrehajtani az utasításokat. Ehhez egy kis programot kell az EPROM-ba írnunk (hogy hogyan, mi módon, azt majd egy másik cikkben írjuk le). Ez a program célszerűen olyan, hogy az élesztés következő lépésében a kimeneti port működőképességét is mutatja. A program az alábbi.

EPROM cím	*Tartalom*	*Megjegyzés*
0000 ED 56	IM1	1 módusú megszakítás
0002 FB	EI	Megszakítás-engedélyezés
0003 3C	UJ: INC A	Az akkumulátor növelése
0004 D3 00	OUT (Ø),A	Kivitel 74LS273-ra
0006 18 FB	JR UJ	

A program az akkumulátor tartalmának állandóan növelt értékét írja ki a kimeneti portra.



4. ábra

Az ezt a programot tartalmazó EPROM-ot a foglalatba dugva, RESET után az A0—A2 címvonalon, a D0—D7 adatvonalakon, az RD, WR, IORQ, MREQ, M1 jelű kivezetéseken jeleket kapunk, és a 74LS273 IC 11. lábán (a kimeneti portot realizáló áramkör beíró bemenete) is láthatók impulzusok. Ezek után beforraszthatjuk a 74LS273, 74LS244 IC-ket. Írjuk az EPROM 66H címére a következő programot:

EPROM cím	*Tartalom*	*Megjegyzés*
0066 DB 00	TR: IN A, (0)	Bemenet beolvasása
0068 D3 00	OUT (0), A	Kivitel 74LS273-ra
006A 18 FA	JR TR	

Ez a kis program akkor kezd működni, ha a Z80 17. lábát (NMI) egy pillanatra nullára kötjük; azt csinálja, hogy a bemeneti port csatlakozóira kapcsolt logikai szintű jeleket a kimeneti portra másolja. Ezt úgy ellenőrizhetjük, hogy a bemenetekre nulla, illetve 5 V-ot kapcsolva, ez a V4 kimenetein invertálva jelenik meg, mert a 74LS244 invertálja a bemeneti jeleket. Ha ez így kész és működik, akkor pici számítógépünk működőképes, lehet benne programot futtatni, bemeneteiről jeleket olvasni, kimeneteire jeleket írni.

Ezek után már csak az a kérdés, hogy mire használjuk a gépet . . .? Az alkalmazásokat bemutató írásokat az ötletes olvasóktól várjuk, és a legjobbakat közölni fogjuk.

Néhány megvalósítható ötletet azért mi is adunk.

1. Nyomógombos ajtózár logikai egysége. Három-négy nyomógombot kötve a bemenetekre, azok megfelelő sorrendű — és esetleg időtartamú — megnyomásával jelenik meg a jel az egyik kimeneten, ami működteti az ajtózár mágnesét. Egy másik hasznos kimenet lehet a riasztójel (néhány sikertelen próbálkozás után).
2. Modellvasút-irányító/vezérlő
3. Morzegyakorló
4. Soros formátumú adatok párhuzamossá alakítása és viszont.

Megjegyezzük, hogy a mikroprocesszoros áramköri lap képes nagyobb kapacitású EPROM-ok (2732, 2764, 27128, 27256 stb.) befogadására is. Ehhez 28 lábú IC foglalatot kell az EPROM-okhoz beültetni, és a J1 . . . J4 átkötéseket megfelelően kell összekötni; ezt itt nem részletezzük.

Akiknek nincs idejük vagy kedvük az építéshez, de mégis szeretnének egy ilyen mikroszámítógépet, azoknak a fent leírt áramkört a szerkesztőségen keresztül megajánljuk: meg lehet rendelni és meg is lehet vásárolni haszonmentes áron a bemutatott hardvert.

— Nyomtatott áramköri lap 300,— Ft
— A komplett mikroszámítógép zacskóban, szereletlenül 1300,— Ft
— A szerelt, működőképes mikrogép, az EPROM-ba beégetett próbaprogrammal 1600,— Ft

DR. KÓNYA LÁSZLÓ

PRIMO

Grafikus nyomtatás K6311 mátrix-nyomtatóval

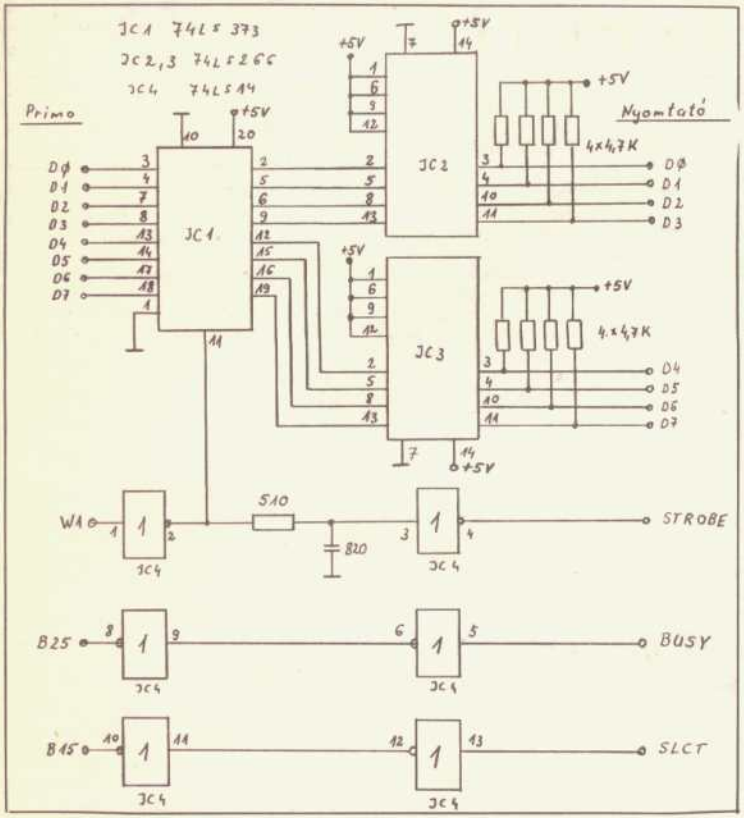
A program a képernyőtartalom grafikus kinyomtatását végzi el. A kapcsolási rajz a PRIMO-t illeszti Centronics típusú interfésszel rendelkező nyomtatókhoz (lásd az ábrát).

A nyomtatóprogram (1. lista) felhasználja a PRIMO grafikus pontvizsgáló rutinját (belépési címe a decimális 137), amely megvizsgálja a D, E regiszterben beállított értéknek megfelelő grafikus képernyőpontot. Ha a pontot aktívnak találja, a ZERO flag értéket 0-ra állítja, ellenkező esetben 1-re (lásd a PRIMO szoftverkönyvét). A nyomtató grafikus üzemmódba állítása után a bal felső sarokból kiindulva megvizsgálja az egymás alatt elhelyezkedő 6 képpontot. A képpontnak megfelelő értéket az A regiszteren keresztül a nyomtatóra küldi. Összefüggő képet a mátrixnyomtatóval csak akkor tudunk előállítani, ha mindössze 6 tűt vezérlünk, és soremelés helyett függőleges relatív pozicionálást végzünk.

A grafikus nyomtatóprogram BASIC változatában (2. lista) a program 20—40.

1. lista

```
                        ORG   6000
                        LOAD 6000
                        PUSH HL
                        PUSH DE
6002 FD218E60           LD    IY,PEKESZ
6006 FD3600BF           LD    (IY+0),191T
600A FD360100           LD    (IY+1),0
600E 3E7F               LD    A,127T
6010 CD9D60             CALL KA
6013 FDE5       EL:     PUSH IY
6015 0604               LD    B,4
6017 FD7E02     EZ:     LD    A,(IY+2)
601A CD9D60             CALL KA
601D FD23               INC   IY
601F 10F6               DJNZ EZ
6021 FDE1               POP   IY
6023 FD360600           LD    (IY+6),0
6027 FD5E01     MA:     LD    E,(IY+1)
602A 0606               LD    B,6
602C FD5600     RA:     LD    D,(IY+0)
602F FDE5               PUSH IY
6031 CD8900             CALL 137T
6034 FDE1               POP   IY
6036 2006               JR    NZ,AT
6038 FDCB06B6           RES   0,(IY+6)
603C 1804               JR    AS
603E FDCB06C6 AT:       SET   0,(IY+6)
6042 FDCB0606 AS:       RLC   (IY+6)
6046 FD3500             DEC   (IY+0)
6049 10E1               DJNZ RA
604B FD7E06             LD    A,(IY+6)
604E FD360600           LD    (IY+6),0
6052 CD9D60             CALL KA
6055 FD3401             INC   (IY+1)
6058 3E06               LD    A,6
```



Nyomtatóillesztő Primóhoz

```
10 REM GEPI KOD-DATA-BA ATALAKITAS
100 CLS
120 DEFINTA-Z
130 INPUT"kezdocim";E:KeA=E
140 INPUT"vegcim";E
150 V=18211
160 POKE V,0:POKEV+1,0
165 V=V+2
255 S=0:T=0:U=0
265 S=S+10:POKE V,S-INT(S/256)*256:POKEV
+1,INT(S/256):POKEV+2,136:POKEV+3,32:V=V
+3
275 I=I+1:U=U+1:A$=STR$(PEEK(K+I)):FOR B
=2TOLEN(A$):U=U+1
280 POKEV,ASC(MID$(A$,B,1)):NEXT
285 IF K+U=EORI=45.305
295 V=V+1:POKEV,44:A$=""
305 I=0:B=V:FOR Z=1TO3:POKEV+2,0:NEXT Z:
V=V+2
315 J=7-(:IFPEEK(7)+PEEK(J-1)+PEEK(J-2)
)=0.315
325 POKEJ,INT(V/256):POKE(J-1),V-INT(V/2
56)*256
335 V=V+2:IF K+U=E.J45ELSE365
345 POKE 16722,V-INT(V/256)*256:POKE1672
3,INT(V/256)
355 POKE16733,PEEK(16722):POKE16734,PEEK
(16723)
365 DELETE 10-375
375 END
385 ********************************
********************************
********************************
********************************
********************************
*********
386 ********************************
********************************
********************************
********************************
********************************
*********
387 ********************************
********************************
********************************
********************************
********************************
*********
388 ********************************
********************************
********************************
********************************
********************************
*********
```

3. lista

2. lista

```
5 REM KEPERNYO TARTALMAT KINYOMTATO PROG
RAM
10 CLS
20 C=PEEK(16561)+256*PEEK(16562)-170
30 POKE 16561,C-256*INT(C/256)
40 CLEAR 50:C=PEEK(16561)+256*PEEK(16562
)+1
50 PRINT"A PROGRAM INDITAS UTAN TORLODIK
.          A FIELD BASIC SZOVAL A KEPERN
YO          TARTALMAT A NYOMTATORA KULD
I"
52 PRINT:PRINT:PRINT"Kezdhetem? (I)"
60 A=C-256*INT(C/256):B=INT(C/256)
65 IF INKEY$="I"OR INKEY$="i"THEN70ELSE 65
70 POKE 16764,195,A,B
72 C1=C+158:C2=C+143
74 D=C1-256*INT(C1/256):E=INT(C1/256)
76 H=C2-256*INT(C2/256):L=INT(C2/256)
80 C=C-65536
100 POKE C,229, 213,253,33,H,L,253,54,0,
191,253,54,1,0,62,127,205,D,E,253,229,6,
4,253,126,0,205,D,E,253,35,16,246,253,22
5,253,54,6,0,253,94,1,6,6,253,86
200 POKEC+46, 0,253,229,205,137,0,253,22
5,32,6,253,203,6,134,24,4,253,203,6,198,
253,203,6,6,253,53,0,16,225,253,126,6,25
3,54,6,0,205,D,E,253,52,1,42,6,253
300 POKEC+91,134,0,253,119,0,253,126,1,2
54,0,40,2,24,190,6,8,253,229,253,126,7,2
05,D,E,253,35,16,246,253,225,253,54,1,0,
253,126,0,254,5,40,7,214,6,253,119
400 POKE C+136,0,24,13a,209,225,201,0,0,
0,27,75,0,1,0,27,91,49,101,27,91,48,96,2
45,219,5,230,8,32,250,241,211,64,201
420 CLS
430 NEW
```

sora a gépi kódú rutin számára foglal helyet a képernyő RAM előtti területén. A program RUN után átírja a RAM végcímét, így a nyomtatórutin a memóriában (a BASIC programtól) védett helyen lesz. A grafikus nyomtatást a Field szóval indítjuk. A BASIC ezt a szót nem használja. A Field szó ugráscímét a 70-es sorban írjuk át. Ahhoz,hogy az assembly program ne csak a 6000-es fordítási címtől működjék, a címek értékét más kezdőcímeknek megfelelően is ki kell számítani. Ez történik a 72—76. sorban.

Végül közöljük azt a segédprogramot (3. lista), amely az assembly tárgyprogramot BASIC DATA sorokba rendezi. A programnak a magazinban megjelent HT-program az alapja (Mikroszámítógép Magazin 86/4. 3. old.).

Az inverz üzemmód az assembly program 24. sorában JR NZ, AT helyett JR z,AT irásával állítható be, vagy a BASIC program 200-as sorában levő 32-es helyett 40-et kell írni.

GERENDÁS SÁNDOR

THE USERS PORT

San Fernando Valley Commodore Users Group

Biztos, hogy hibás a számítógépünk?

Ha úgy látjuk, hogy elromlott a számítógépünk, kövessük az alábbi tanácsokat. Biztosan megoldják minden problémánkat . . .

1. Határozottan lépjünk fel a betegeskedő géppel szemben. Ekkor azt fogja hinni (gyakran tévesen), hogy tudunk valamit. Ha történetesen valaki figyel is bennünket, meg lesz győződve arról, hogy mi javítottuk meg a masinát. De ha mégsem sikerül, nézzük a második tanácsot.

2. Lobogtassunk egy műszaki kézikönyvet a számítógépnek. Így elhitetjük vele, hogy legalább azt tudjuk, hol kell utánanézni a dolgoknak. Ha ez is hiábavaló, akkor a harmadik tanács szerint járjunk el.

3. Erélyesen idézzük a gépnek az Ohm-törvényt. (Előbb azonban biztos helyen nézzünk utána!) Ez elég drasztikus lépés. Ha a számítógép még mindig nem működik, következzen a negyedik tanács.

4. Rázzuk meg enyhén a gépet. Ehhez három-hat erős lábdobbantásra van szükség, lehetőleg betonpadlón. Ez is drasztikus lépés. Amennyiben ez sem vezetne sikerre, fogadjuk meg az ötödik tanácsot.

5. Hozzunk egy IC chipet. Ez azt bizonyítja, hogy járatosak vagyunk a számítógép-tervezésben, és növeli fölényünket a géppel szemben. De akkor sincs baj, ha ez sem válik be, ugyanis ez esetben BIZTOSAN JÓ A SZÁMÍTÓGÉP. Ekkor kell megszívlelnünk a hatodik, legdrasztikusabb tanácsot.

6. OLVASSUK EL A SZÁMÍTÓGÉP KÉZIKÖNYVÉT!!!

(A „The Users Port" cikke alapján)

PÁL MAGDA

Két érdekes rövid program C64-re

LOAD.

Ez a rutin programfájlhoz kapcsolva, azt tetszés szerinti tárcímre tölthetővé teszi. A program 73 bájtot foglal el, és bárhol lehet, ahol ennyi hely van. A 251—3 címeket azonban használja, így ott ne helyezzük el! A rutint az 50-es sorban levő cimtől tölti; a kívánt kezdőcím ide írandó be. A használati forma a 40-es sorban látható.

```
LOAD.ILLIST
10 REM LOADS A PRG AT A GIVEN ADDRESS
20 REM 100% RELOCATABLE-EDIT LINE 50
30 REM SYNTAX:
40 REM SYS A,"FILENAME",DEVICE NUMBER,LO
AD ADDRESS
50 A=49152
60 FOR I=A TO A+72:READ B:POKE I,B:NEXT
70 END
80 DATA 032,253,174,032,158,173,032,143
90 DATA 173,169,100,160,101,032,219,182
100 DATA 160,002,177,100,153,251,000,136
110 DATA 016,248,165,251,166,252,164,253
120 DATA 032,189,255,032,253,174,032,158
130 DATA 173,032,247,183,152,170,169,008
140 DATA 160,000,032,186,255,032,253,174
150 DATA 032,158,173,032,247,183,072,152
160 DATA 170,104,168,169,000,032,213,255
170 DATA 096
```

FREAKOUT

Igen furcsa hatású program: a regiszterek tartalmát áttölti a SID IC-be, amikor megszakítást ad. Egy hosszabb programot listázva, a CTRL gomb lenyomására érdekes hangokat kaphatunk. A POKE 56325,1 vagy POKE 56324,3Ø:POKE 56325,Ø beadására a megszakítás frekvenciáját változtatjuk, ezért változik a hangmagasság is.

SIMONYI ZSUZSA

```
FREAKOUT
90 REM FREAKOUT
100 S=54272:L=49152:POKE S+24,15
110 FOR I=S+4 TO S+18 STEP7:POKE T+2,240
:POKE T,17:NEXT
120 FOR I=L TO L+21:READ A:POKE T,A:NEXT
130 POKE 56334,PEEK(56334)AND254
140 POKE 789,L/256:POKE 788,L-PEEK(789)*
256
150 POKE 56334,PEEK(56334)OR1
160 DATA186,189,1,1,141,1,212,189,2,1
170 DATA141,8,212,189,3,1,141,15,212,76,
49,234
```

Elektronikus lemez

Az idei év hazai piaci újdonságának számítanak az elektronikus lemezek, melyek tulajdonképpen lemezes perifériaként kezelt félvezetős tárak. Az alkalmazásuk csupán másodlagosan jelent háttértárbővítést, ennél sokkal jelentősebb az üzemeltetési paramétereknek a sokszor nagyságrendekben mérhető javulása: felgyorsul az adatelérés, csökken a feldolgozási idő, megnő a rendszer átbocsátó képessége. Mivel így az elektromechanikus lemezegységek terhelése kisebb, javul a számítógépes rendszer műszaki megbízhatósága. A fejmozgások és a forgó mozgás elmaradása, valamint a közvetlen hozzáférés, továbbá a nagy sebességű párhuzamos adatátvitel következtében az elektronikus lemez egyesíti magában a központi memóriákra jellemző megbízhatóságot és teljesítőképességet a mágneslemezes tároknál megszokott kezelési kényelemmel.

Az elektronikus lemezben rejlő előnyök különösen jól érvényesíthetők a következő területeken:

— Sok fejpozicionálást igénylő, nagyméretű adatbázis-kezelő rendszerekben. Amennyiben srukturált adatbázissal dolgozunk, érdemes a program és az adatok minden elemét elektronikus lemezre tenni. Ha ehhez a rendelkezésre álló elektronikus lemezkapacitás kicsi, akkor elegendő a leggyakrabban használt elemeket, például az indextáblákat az elektronikus lemezre áthelyezni, s a konkrét adatok maradhatnak a mágneses perifériákon is.

— Programok szegmentálásánál a vezérlőprogramot célszerű az elektronikus lemezre helyezni.

— Nagyméretű munkaállományokat az elektronikus lemezre téve jelentősen gyorsulhat a feldolgozás.

— A mátrixműveletek igen memóriaigényesek. A mátrixok egy részét a feldolgozás során háttértárra (elektronikus lemezre) kell kitenni.

Az elektronikus lemez hatékonyságnövelése jól kihasználható még a grafikus feldolgozásoknál, a távfeldolgozásnál, az interaktív rendszerekben, a folyamatszabályozásnál stb.

Az elektronikus lemezek hazai választékáról szólva kezdjük azzal, hogy az eddigiekben vázolt előnyök a nagygépes feldolgozásoknál nyilvánvalóak. A PDP-kompatibilis hazai és szovjet gyártmányú gépekre, továbbá az R—10-re az elektronikus lemez már elkészült, s az Alkotó Ifjúság Egyesülés forgalmazza. 256 kbájttól 32 Mbájtig terjedő elektronikus lemezkapacitás építhető ki 1 és 4 Mbájtos, valamint 256 kbájtos modulokból. Az ár félmillió forintig terjed.

Az IBM PC-vel kompatibilis gépekhez 2 Mbájtos elektronikus lemezt ajánl a Microsystem Kisszövetkezet 180 ezer forintért.

A Commodore 64-re a Coopker—Summatech szakcsoport kínál elektronikus lemezt, 256 kbájt kapacitással. Ezt egy 1541-es hajlékonylemezes tárként lehet kezelni, vagyis hagyományos lemezt használó programok az elektronikus lemezzel is változtatás nélkül futnak. A lemezműveletek felgyorsításával a Commodore 64-en futó adatfeldolgozó programok futási ideje jelentősen csökken.

Az elektronikus lemezek gyakorlati alkalmazását a félvezető tárak árának radikális csökkenése, tárkapacitásának pedig jelentős növekedése tette lehetővé.

BITEK ÉS FIGURÁK

Ütéscsere-algoritmus

Az eljárás célja annak meghatározása, hogy a soron következő játékos ütéssel elérhet-e anyagi előnyt, másképpen: egy ütéssorozat végén melyik fél kerül ki győztesen a csatából. Bizonyosra ezt minden lehetséges ütésváltás feltárásával lehetne meghatározni, de a megvizsgálandó változatok száma igen nagy, így ez a módszer reménytelenül hosszadalmas lenne.

Az ütéscsere-algoritmus elkerüli minden változat elemzésének a szükségességét, mégis ugyanarra az eredményre vezet. A módszer azon alapszik, hogy felírjuk egy bizonyos mezőre a támadó és a védő figurák listáját (amelyik fél lépésre következik, az a támadó). Mindkét listát növekvő érték szerint rendezzük, kivéve akkor, ha egy figura egy másikon keresztül támad. Az *1. ábrán* látható álláson jól szemléltethetjük a támadó és védekező fél listáját a d5 mezőre.

A támadó lista: Hc3, Fg2, Vd2, Vd1

A védekező lista: Hb6, Fb7, Bd8.

A gyakorlatban két báb cseréje óriási különbséggel jár. Ha a d1 mezőn álló vezért felcseréljük a d2-n álló bástyával, akkor a támadó lista a következőképpen alakul: Hc3, Fg2, Vd2, Bd1.

Az ütéscsere-algoritmus kiszámítja az éppen lépő, a vizsgált mezőt támadó játékosokra az értékeket. Amikor a mezőn valamilyen báb áll, akkor az algoritmus kiszámítja a megtámadott mezőt érő ütés értékét.

Egy mező ütéscsereértéke az a lehető legtöbb nyereség, amit a lépő félnek e mezőt érő ütéseinek, vagyis cseréinek végigkísérésével kapunk úgy, hogy az ellenfél a legkisebb anyagveszteségre törekszik.

Az eljárás célja, hogy el lehessen dönteni, érdemes-e ütni a változatok sokaságának továbbszámolása nélkül. Az eljárással meghatározhatjuk egy üres mező elfoglalásának biztonságosságát is.

Az ütéscsereérték számításának az alapja, hogy a lépésen levő fél vagy egy védtelen ellenséges figurát üt, vagy pedig a legkisebb értékű támadó bábjával üt úgy egy védett ellenséges figurát, hogy visszaütés esetén az ellenfél a lehető legkevesebbet nyerjen. Tegyük fel, hogy a V_0 értékű világos figurát n darab sötét figura támadja, T_1, T_2, T_3, . . ., T_n értékkel, érték szerint növekvő sorrendben, és n darab világos figura védi a mezőt V_1, V_2, V_3, . . ., V_n értékekkel, szintén nagyság szerint rendezve. Ha sötét lép, a legkisebb értékű figurájával üt és ha világos nem üt vissza, akkor sötét $W_1 = V_0$ összegű anyagot nyer. Ha világos ezt visszaüti, és sötét nem folytatja az ütést, akkor sötét nyeresége $V_0 - T_1$ lenne. Ha viszont sötét üt, világos visszaüt, sötét harmadszor is üt, és világos nem folytatja az ütések sorozatát, sötét tiszta nyeresége $W2 = V0 - T1 + V1$. Így egy bizonyos mezőn az ütések sorozatát meg tudjuk határozni, nem számít, hogy az ütések sora mikor fejeződik be.

Tehát:

$W_1 = V_0$

$W_2 = V_0 - T_1 + V_1$

$W_3 = V_0 - T_1 + V_1 - T_2 + V_2$

$W_4 = V_0 - T_1 + V_1 - T_2 + V_2 - T_3 + V_3$

.

.

.

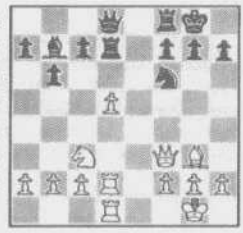
és

$B_1 = V_0 - T_1$

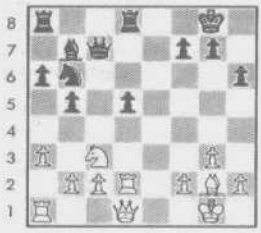
$B_2 = V_0 - T_1 + V_1 - T_2$

$B_3 = V_0 - T_1 + V_1 - T_2 + V_2 - T_3$

$B_4 = V_0 - T_1 + V_1 - T_2 + V_2 - T_3 + V_3 - T_3$



1. ábra



2. ábra

Ezt a két sort addig kell kiszámolnunk, amíg az egyik fél úgy dönt, hogy nem folytatja az ütések sorát.

Két oka lehet az ütések abbahagyásának. Vagy az, hogy az egyik fél, bár rendelkezik az ütés lehetőségével, mégis úgy dönt, hogy nem üt többet, vagy mindkét fél addig folytatja az ütést, amíg nincs több figura, amivel folytatni lehetne az ütéssorozatot a kérdéses mezőn. Sötét a lista páros számú elemeinél állhat meg — világos ütése után —, világos pedig a páratlan indexű elemnél — sötét ütése után.

Aki először üt — esetünkben sötét *(2. ábra)* —, megpróbálja elérni a páros indexű elemek maximumát, mivel minden páros indexű elem után sötétnek megvan a lehetősége az ütés abbahagyására. Amelyik fél másodiknak üt — esetünkben világos —, az pedig a páratlan indexű elemek minimumát próbálja elérni, mivel világosnak a páratlan indexű elemek után van lehetősége az ütések sorát megállítani.

Tekintsük például a következő állást:

Kg1, Vf3, Bd1, Bd2, Hc3, Fg3, a2, b2, c2, f2, g2, h2, d5

Kg8, Vd8, Bf8, Bd7, Hf6, Fb7, a7, b6, c7, f7, g7, h7

Ha a d5 mezőt vizsgáljuk, a következő eredményre jutunk:

$V_0 = 1$ világos gyalog

$V_1 = 3$ világos huszár

$V_2 = 5$ világos bástya

$V_3 = 5$ világos bástya

$V_4 = 9$ világos vezér

$T_1 = 3$ sötét futó

$T_2 = 3$ sötét huszár

$T_3 = 5$ sötét bástya

$T_4 = 9$ sötét vezér

A támadó és védekező figurákat figyelembe véve a következő értékeket kapjuk:

$W_1 = 1$

$B_1 = -2$

$W_2 = 1$

$B_2 = -2$

$W_3 = 3$

$B_3 = -2$

$W_4 = 3$

$B_4 = -6$

Ebből a sorozatból alkossunk olyan módon két részsorozatot, hogy az egyikben csak a W értékek szerepeljenek, a másikban pedig csak a B értékei. Az eredeti sorozat utolsó értékét mindkét részsorozat elemeként vegyük figyelembe. Ezt tekinthetjük úgy is, mint utoljára egy 0 értékű ütést. Ezek alapján a két részsorozat: $(-2, -2, -2, -6)$, amelynek maximuma: -2

(1,1,3,3,−6), amelynek minimuma: -6

Sötétet a második helyen elért -2 érték után nem lehet a folytatásra kényszeríteni, így világos számára a -6 érték elérhetetlen. Tehát a listát a második elem után meg kell nyesni, és az utolsó elemet mindkét részsorozatban figyelembe kell venni, mint már tárgyaltuk.

Ezek után sötét ismét megpróbálja a páros indexű elemek maximumát elérni, ami esetünkben -2. Világos pedig a páratlan indexű elemek minimumára törekszik, ami szintén -2. Mivel sötét maximuma megegyezik világos minimumával, ezért a folyamatot befejezettnek tekinthetjük, és megállapíthatjuk, hogy ha sötét ezen a mezőn ütést kezdeményez, akkor két gyalogegyenértéket veszít. Tehát sötétnek nem szabad ezen a mezőn ütést kezdeményeznie.

Egy csereanalízis nélküli sakkprogram ezt csak egy nyolcrétegű kutatás után tudná megállapítani. Ezért nagyon fontos az ütésérték-számítás.

A cserealgoritmusnak két fontos korlátja van, ami növelheti a hiba lehetőségét. Nem veszi figyelembe az olyan helyzetet, amelyben

a) *egy figura kötésben áll*, és ezáltal lehetséges, hogy több védőt vagy támadót vesz számításba, mint a valóságban;

b) *ha egy-egy figura túlterhelt* — vagyis azt az esetet, ha egy figura egyszerre két védekező funkciót lát el.

Az első korlátot kiküszöbölhetjük úgy, hogy amelyik figura kötésben áll, azt eleve nem vesszük figyelembe. Ebbe a kategóriába csak azok a figurák tartoznak, amelyek saját királyuk miatt állnak kötésben.

Mivel ez az ütéscsere-algoritmus sem pontos, csak statisztikai adatokat ad, ezért nem célszerű az összes W és B értéket kiszámítani, ami lassítja a programot. Egy gyorsabb eljárással legtöbbször azonos eredményre jutunk, és sok időt megtakaríthatunk vele.

Előnye, hogy nem kell az összes támadó és védő figurát „megjegyezni", hanem elég csak mindig a legkisebb anyagi értékűt, így megtakarítunk egy sorba rendezést. Csupán azt kell tárolni, hogy a leütött figurát védik-e, és ha igen, akkor a támadók vagy a védők vannak-e többen. Ezt a legcélszerűbb egy változóként kezelni, amelynek értéke zérus, és annyit adunk hozzá, ahányan védik, illetve annyit vonunk le, ahányan támadják; ha viszont nem védi semmi, akkor értéke egy extremális elem.

Mindezt figyelembe véve a következőképpen határozhatjuk meg az ütésértéket:

1. Ha nincs védve, akkor az üthető figura értéke.

2. Ha védve van és nem lehet visszaütni, akkor: max. (üthető, üthető — legkisebb támadó)

3. Ha védve van és vissza lehet ütni: min. (üthető — legkisebb támadó + legkisebb védő).

Ha egy teljes állás ütésértékét akarjuk kiszámítani, akkor az előbbi eljárást minden mezőre el kell végezni, minden mezőre külön-külön megállapítani az ütésértéket, és ezután venni ezek maximumát, majd ugyanebben az állásban ki kell számítani minden mezőre az ütésértéket úgy, mintha most lépne az ellenfél, és az így kapott értékek közül a második legnagyobbat kell venni. Ennek a két értéknek a különbsége adja az állásban a lépő fél legnagyobb ütésértékét.

Gondoljuk át, miért is kell kiszámítani az ellenfél szemszögéből is ugyanerre az állásra az ütésértékeket, és miért a második legnagyobbat kell venni ezek közül.

Az ellenfél szemszögéből azért van rá szükség, mert amikor léptünk, átadtuk a lépés jogát, és az ellenfél anyagot nyerhet, ha ezt nem vesszük időben észre.

Azért kell viszont a második legnagyobb ütésértéket figyelembe venni, mert nagyon valószínű, hogy az aktuális lépésünkkel az ellenfél legnagyobb ütésértékű lépését meghiúsítjuk.

Az ütésérték számítására alkalmazott eljárások sajnos nem pontosak, de vegyük figyelembe, hogy a sakkprogramok általában 3 féllépésig minden változatot átszámítanak, tehát az 1., 2., 3. féllépésnél számított ütésérték hibáit kiküszöbölik, a mélyebb kutatások eredményei pedig zömükben statisztikai adatok, így nagyon pontos számításra nincs is szükség. Fontosabb, hogy minél több változatot vizsgáljon át a program, mert a vizsgálatok a statisztikai hibákat kiküszöbölik.

x x x

A sakkprogramozás alapfogalmainak ismertetését ezen a helyen — mivel időközben a szerzőnek *A sakkprogramozásról mindenkinek* c. könyve is megjelent a Novotrade Rt. kiadásában — fentiekkel befejezzük. A továbbiakban a sakkalgoritmusokkal kapcsolatos, egyes olyan speciális problémákkal foglalkozunk, amelyek részletes kifejtésére eddig sem itt, sem a könyvben nem volt lehetőség, illetve a világban folyó kutatások és a szerző saját vizsgálódásai során újabban merültek fel.

KOVÁCS P. ATTILA

KANYAR a MARS-on

A térképet és a szöveget házasította össze sikeresen egy egységesen kezelhető adatbázisrendszerben az Alkalmazástechnika Kisszövetkezet. A térképpel segített objektum-nyilvántartó rendszer (Map Aided Registration System = MARS) célja az adott földrajzi helyhez köthető objektumoknak a térképen való elhelyezése, illetve az adott azonosítóval rendelkező objektum grafikus térképkörnyezetének és szöveges címének a megjelenítése az objektum egyéb szöveges adatai mellett.

Lehetőség van az adatbázisból szöveges kritérium alapján történő visszakeresésre, azonosító szerinti visszakeresésre, sőt egy adott térképszelvényen a kurzor mozgatásával is vissza lehet keresni.

A térképi és a szöveges információ egységes kezelése teljesen új, rendkívül széles távlatokat nyit a számítástechnika-alkalmazásban. Csak vázlatszerűen néhány lehetőség: az országúti hálózaton elhelyezett objektumok nyilvántartása; a nagy raktárakban az egyes árufajták elérése és nyilvántartása; kiállítási pavilonok elhelyezkedése és elérése; kórházak berendezése és a kórházi berendezések nyilvántartása.

Az általános célú MARS rendszer első konkrét megvalósítása is megkezdődött. Ez a Fővárosi Tanács számára készülő *K*özlekedési *A*dat*ny*ilvántartó *A*lap*r*endszer, azaz KANYAR, melynek feladata a főváros térképének egy-egy konkrét földrajzi pontjához kötött közlekedési objektumok és események nyilvántartása azok optikai és verbális jellemzőivel, a nyilvántartott adatok lekérdezése, módosítása az igények alapján. Ennek során megvalósul a konzisztens követés

— a főváros grafikus térképének,

— a térképen elhelyezett objektumok és események azonosításának és szöveges adatainak, továbbá

— egy adott térképpont szöveges címeinek (kerület, utca, házszám) alapján,

függetlenül attól, hogy a fenti háromból melyik változott meg utoljára. A kialakítás alatt álló rendszert bonyolítja, hogy az adathalmaz felett több intézmény rendelkezik, mindegyiknek megvan a saját intelligens terminálja, magát az adatbázist egy központi gépen tárolják. Az adatbázis bármely részét bármely intézmény lekérdezheti, de a módosítási jog szigorúan fel van osztva az intézmények között.

A MARS rendszer az IBM PC-vel kompatibilis gépekre készült. A konkrét megvalósításnál, a KANYAR-nál a terminálok szerepét egy-egy XT teljesítményű gép tölti be, a központi gép feladatait pedig egy 32 bites megamikro számítógép végzi.

„Aprócska" játékos trükkök, melyek szórakoztatnak és egy picit a matematikai ismereteket is feleleventik.

Petike szétszedi apuka óráját

Petike rosszalkodott: fölnyitotta apukája óráját. Leszedte a kis- és nagymutatót, de az óra szerkezetéhez egyébként nem nyúlt. Közben apuka befejezte a kertészkedést, belépett a házba; Petike gyorsan visszaraktа a mutatókat, de véletlenül felcserélte őket. Bár rájött erre, nem maradt idő a helyesbítésre. Apuka karjára tette az óráját, és így szólt:

— Nahát, pontosan 12 óra van, indulnom kell!

Petike lapított, de apja távoztával továbbra is komolyan aggódott; észreveszi-e apuka, hogy a mutatók nincsenek a megfelelő helyen. Például már egy óra múlva is kész lebukás fenyegeti, hiszen a nagymutató egy rendes órán nem állhat az egyesen, amikor a kismutató 12-t mutat.

A srác szerette a matematikát — ez nem áll antagonisztikus ellentétben a felnőttek által ráragasztott minősítéssel: Te rosszaság! —, és szenvedélyesen foglalkozott mindennel, aminél használhatta a számítógépét. Kíváncsi volt, hogy mennyi az esélye annak, hogy a papa fölfedezi a csínyt. A következőképpen gondolkodott: először ki kell derítenem, hogy milyen helyzetekben nem veheti észre az öreg az óramutatók felcserélését; ha kiszámolom az összes mutatóállást, akkor már csak válogatás kérdése, hogy megtudjam, melyek a kedvező együttállások, és mennyi a lebukás veszélye.

Petike munkához látott, s a többféle lehetséges megoldás közül a következőt választotta. (Természetesen ez nem „menti" fel az olvasót az alól, hogy más megoldást is keressen.) Algebrai egyenleteket írt fel, és azok segítségével oldotta meg a kérdést. Közben eszébe jutott Newton mondása, amit az algebrakönyvben olvasott: „Hogy egy számokra, illetőleg elvont mennyiségekre vonatkozó kérdést megoldjunk, nem kell mást tennünk, mint lefordítani a feladatot anyanyelvünkről az algebra nyelvére" (J. A. I. Perelman: Szórakoztató algebra. Bp., 1975. Gondolat, 41. old.).

A nagymutató és a kismutató különböző sebességgel mozog. Ezt úgy is mondjuk, hogy egy óvegységet más idő alatt járnak be. A megoldás során a mutatók járását — a metamatika szabályait betartva — összehasonlíthatóvá tesszük. Egy kicsit pontatlanul fogalmazva — elnézést kérek a matematikusoktól — közös nevezőre kell őket hozni. De a legfontosabb, hogy az algebra nyelvén le kell írni, hogy mikor mutat az óra „értelmes" időt.

A kismutató, illetve az azt mozgató tengely egy óra alatt öt óvegységet halad. Jelöljük a kismutató által megtett óvegységet x változóval. Speciálisan ez megfelel öt percnek is. Így x algebrai változó két fogalmat is jelöl egyszerre: óvegységet és időt is. A nagymutató esetében az elmozdulást jelöljük y változóval. A változók mértékegysége óvegység, illetve perc.

Írjunk fel olyan egyenleteket, amely kifejezi az óramutatók „értelmes" állásait. Vezessük le az egyenletet. Például 1 órakor a nagymutató a 12-es számon áll. Belátható, hogy egy óra, azaz a nagymutatót mozgató tengely egy körbefordulása alatt a kismutató, illetve tengelye öt óvegységet tesz meg. Ha a „körbejárásokat" egyenlővé tesszük, akkor egyórányi időtartamra $x = 5$, azaz

$$\frac{x}{5} = 1$$

A nagymutató esetében ez $y = 60$, azaz

$$\frac{y}{60} = 1$$

Az egyenlőség a nagymutató egy körbejárását írja le. Sikerült tehát a mutatók elmozdulásait „közös nevezőre hozni" azzal, hogy óránként való elmozdulásaikat normáltuk.

Belátható, hogy az

$$\frac{x}{5} - \frac{y}{60} = m$$

algebrai egyenlet szerint meghatározott m 0-tól 11-ig terjedő egész értéket vesz fel. Aki nem gondolja át a feladat megfogalmazását, azt mondhatja, hogy x és y bármely értéket felvehet; így egyáltalán nem biztos, hogy m egész érték! Miért fejezi ki ez az egyenlet az óra helyes járását, és miért egész szám m 0 és 11 között? A fenti egyenlet önmagában természetesen nem biztosítja ezt. De ha az alapfeladatot értelmezzük, és arra vonatkozó megoldóegyenletet írunk fel, akkor már más a helyzet. Hiszen x és y egymástól függetlenül nem vehet fel értéket, mint ahogy az egyik mutató sem járhat a másik nélkül. Például, ha fél óra telt el 12 órától, akkor a kismutató $x = 2{,}5$ óvegységet haladt, és mivel a nagymutatót nem állította le senki, így az $y = 30$ óvegységet, azaz percet tett meg.

$$\frac{2{,}5}{5} - \frac{30}{60} = 0$$

Ha ugyanis m nem egész érték, akkor az óra nem mutathat helyes időt. És ez fordítva is igaz. Az m ún. paraméter, amely azt mutatja, hogy hányadik egész órán belül állapítjuk meg a mutatók „értelmes" állását. Az $m = 0$ a 12 órától számított első órán belüli helyes mutatóállásokat teszi meghatározhatóvá.

Az x és y értékét, amint látni fogjuk, egy egyenletrendszer felállításával és megoldásával tudjuk kiszámolni. Mivel x és y függő változók, így értékükre kizáró feltételt nem kell felírni. Egyébként értékük $0 < x; y < 60$ intervallumban. Amikor x vagy y hatvannal egyenlő, akkor nyilván egész órát kell figyelembe venni.

```
5 LPRINT "ORA","PERC"
10 FOR M=0 TO 11
20 FOR N=0 TO 11
30 X=60*(12*M+N)/143
40 Y=60*(12*N+N)/143
50 LPRINT N,X
60 LPRINT M,Y
70 NEXT N
80 NEXT M
90 END
```

Írjuk fel most a mutatók felcserélése utáni helyzetet. Az előző egyenletből kiindulva x-et és y-t fordítva kell normálnunk, hiszen most a kismutató forog gyorsabban a nagymutatónál. Így az egyenlet:

$$\frac{y}{5} - \frac{x}{60} = n$$

ahol n az előbbi egyenletnél leírt logika szerint 0-tól 11-ig terjedő egész szám. Most már két egyenletünk van két ismeretlen meghatározásához. Ha az egyenletrendszert megoldjuk az adott intervallumon belül felvett m és n értékekre, akkor megkapjuk, hogy m, illetve n-edik órában hányadik percben (x és y) mutat az óra értelmes időt.

Az

$$\frac{x}{5} - \frac{y}{60} = m$$

$$\frac{y}{5} - \frac{x}{60} = n$$

egyenletrendszer megoldása x-re és y-ra azért mutatja a helyes megoldást, mert csak akkor teljesül, ha értelmezhető helyen vannak a mutatók.

Az egyenletrendszert például a következőképpen oldhatjuk meg. Hozzuk közös nevezőre az egyenleteket, és a másodikat szorozzuk meg 12-vel:

$$\frac{12x}{60} - \frac{y}{60} = m$$

$$\frac{12*12y}{60} - \frac{12x}{60} = 12n$$

Adjuk össze az egyenleteket:

$$\frac{143y}{60} = m + 12n$$

azután oldjuk meg y-ra és x-re:

$$x = \frac{60\,(12m + n)}{143}$$

$$y = \frac{60\,(12n + m)}{143}$$

Mivel m órához y perc tartozik és fordítva, n órához x perc, így például $m = 6$ és $n = 7$ esetben

$$x = \frac{60\,(12*6 + 7)}{143} \approx 33{,}15$$

$$y = \frac{60\,(12*7 + 6)}{143} \approx 37{,}76$$

A felcserélt mutatók 7 óra 33,15 perc és 6 óra 37,76 perckor mutatnak „értelmes" időt.

Ha Petike m-et és n-et 0-tól 11-ig tartó egész számokkal helyettesíti be, megtudja, mikor mutat apuka órája értelmes időt. Ez 143 esetben fordul elő. Az óra összesen – az azonos állást levonva – $12*60 - 1 = 719$ időpontot mutat. Petike esélye

$$\frac{143}{719} = 0{,}20,$$

azaz 20%, hogy apuka nem veszi észre csínytevését.

Petike kíváncsi volt, hogy milyen időpontokban mutat apuka elrontott órája hihető időt. Mivel 143-szor az egyenletrendszert nem volt kedve megoldani, ezért rövid programot írt számítógépére és kinyomtatta az eredményeket. Az M és N változó az órát, az X és Y változó a percet tartalmazza. A program 144 megoldást ír ki, melyből egy azonos, de nem érdemes emiatt komplikálni.

PINKE GYÖRGY

Bucsu Gyula, Szolnok,

Szántó krt. 33. II. em. 7. 5000

Egy Philips VG 8020 típusú, MSX rendszerű számítógépem van.

Kérném Önöket, ha tudnak olyan klubról, ahol ilyen rendszerű számítógépekkel is foglalkoznak, akkor adják meg a címet. Szolnokon ugyanis nem működik MSX rendszerű gépeket használó klub.

Mi tagadás, mi sem tudunk létezéséről. Viszont nagyon reméljük, hogy e levél közreadása után többen is jelentkeznek. A közvetítést vállaljuk, s ha egymásra találnak, akkor indíthatnának akár egy levelező klubot is az MSX rendszerű géphasználóknak.

ifj. Pálinkás László, Monori-erdő,

Nefelejts utca 45. 2213

Noha kiváló programokat olvashatunk a Magazinban, sajnos azonban a Commodore Plus/4-es számítógéphez az ez évi számokban nem találtam anyagot, pedig minél többet szeretnék e típusra írtakból átvenni. Érdekelne az is, hogy hol kapható ilyen számítógép . . .

A Commodore Plus/4-et a Novotrade 2C áruház forgalmazza. Ennek címe: Budapest XIII., Balzac utca 35. És reméljük előbb-utóbb a C Plus/4-re is kellő mennyiségű programot olvashat lapunkban.

Tukszár György, Nyergesújfalu,

Eternit telep Cső utca 16/1. 2536

Az 1986. októberi számban találtam rá a Szellemgrafika 1. írásra. Az első néhány sort olvasva a programozás élvezetének adtam át magam. Egy óra múlva izgatottan pötyögtem már a RUN-t, de csalódtam, mert a ?OUT OF DATA ERROR kijelzést kaptam. Erre azután az utolsó DATA sorba beírtam néhány nullát. Újra indítottam a programot, — a program vesztére. Letörölte saját magát! Így tehát alaposabban tanulmányoztam az írást, míg az utolsó mondat mindent megvilágított: „Begépelés előtt toljuk el a BASIC terület kezdetét 4 k-ra az ismert módon." Igen ám, de nem ismerem ezt az eljárást, tehát nem tudom végrehajtani vele a programozást. Mindmáig nem találtam senkit aki útba igazított volna. Hiába is rendeltem számos könyvet, valamennyi szükszavúan foglalkozott a grafikai lehetőségekkel. Hát ezért fordulok végül segítségért Önökhöz.

Szeretném, ha megírnák a 4 k BASIC területet eltolásának módját, illetve ha megadnák azoknak a könyveknek a listáját, amelyek kimondottan grafikával foglalkoznak.

Ugyanezen lapszámunk 5. oldalán jelent meg a „Programönkritika is . . ." című írás, melyből ha elolvassa pontos választ kap első kérdésére. A másodikhoz csak azt tudjuk tanácsolni, hogy akár visszamenőleg is, de olvassa el könyvismertetéseinket. Ezekből ugyanis bőséges képet kaphat a grafikai lehetőségeket tárgyaló könyvekről.

Pécsi Richárd, Balassagyarmat,

Rákóczi út 44. 3. em. 13. 2660

A „Csipp-csepp" programmal van problémám. Ebben ugyanis találtam néhány hibát, melyek sajnos a program látványosságát „koptatták".

Az első az 5500-as sorban van. Önök itt a két idézőjel közé egy SPACE-t tettek, holott helyesen egy □■ karakter szerepel. Ugyanez a hiba csúszott a 9010-es sorba is, ahol szintén □■ karakter szerepel a két idézőjel között. Ez a két apróság viszont jelentősen befolyásolja a program kiállítását. De felfedeztem még egy hibát a 1010-es sorban: LET=INT/RND * 25) 1-et írtak Önök, holott RND * 26 lett volna a helyes program.

Igazán köszönjük a korrekciót! Reméljük mások is hasznát veszik a javításnak.

Szalajki Zoltán, Budapest,

Csontváry krt. 27. 1181

Atari 800 XL-em van. Szeretnék turbó programot kapni számítógépemhez. Mindehhez jó lenne tudni az Atari szekció címét is.

Turbó programot nem lehet jelenleg üzletben vásárolni, viszont szívesen közreadjuk a szekció címét. Nevezetesen a HCC keretén belül működik, a Budapest V., Báthori utca 16. szám alatt egy klub, minden hónap első keddjén tartja összejövetelét, délután 17 órakor.

Kardos Tamás, Budapest,

Jókai utca 26. 1066

Szeretnék a Forth programnyelvről terjedelmes cikket olvasni. Nagyon jó lenne, ha mihamarabb ehhez segítséget adnának.

Lapunk 1986. márciusi számában írtunk a Forth programnyelvről cikket. Ha figyelmesen visszafelé tanulmányozza magazinunkat, kérdésére részletes válasszal szolgálhatunk.

Juhász Béla, Nagykőrös,

Ifjúság u. 17/5. III. ker. 2750

Örülnék, ha az Atari 800 XL típusú gépemre kapnék Önöktől programokat, vagy ha ehhez a mikroszámítógépemhez partnerekre lelnék.

Lapunkban eddig még nem közöltünk ilyen programokat, de mivel erre igény van, a jövőben odafigyelünk . . . a címet pedig közreadjuk. Reméljük, talál partnereket . . .

Commodore 64 gépi nyelvű programozásának gyakorlata
A 6510-es μP
Szerk. Bobula András
(Budapest, 1986. IPIK MECHATRONINFO, 223 oldal.
Ára: 290,— Ft.)

A hazánkban egyik legelterjedtebb személyi számítógép, a Commodore 64 felhasználói közül a számítástechnikában kevésbé járatosakat a sorozat előző kötetei a BASIC nyelv használatára tanították meg.

Ez a könyv az igényes számítógépes megoldások alkalmazóinak szól, akiket megismertet a gépi nyelvű programozás alapfogalmaival, példákkal is illusztrálva, hogy mikor célszerű ezt a programozási módot alkalmazni. Majd ismereti a C64 műveletvégző és vezérlő egységét, a 6510-es mikroprocesszort. A processzor utasításkészletének példaprogramokon keresztüli bemutatását a gépi nyelvű program gépbe viteli eszközeinek leírása követi. A kötet utolsó fejezetei példaprogramokat és az új ismeretek alkalmazását elmélyítő feladatok megoldásait tartalmazzák. A függelékben az olvasó a 6510-es μP utasításkészletének táblázatait találja.

Ury László:
Commodore 128. 1. kötet
BASIC és felhasználói kézikönyv + JANE (Budapest, 1987. LSI ATSZ, 295 oldal.
Ára: 297,— Ft.)

A kötet azoknak készült, akik Commodore 64 vagy Commodore 128 személyi számítógéppel rendelkeznek, továbbá azoknak, akik feladataik elvégzésére vagy egyszerűen a maguk szórakoztatására ilyen gépet kívánnak beszerezni.

A Commodore 128 valójában három különböző gép, mivel három lényegesen eltérő üzzemmódban használható. Ezek a Commodore 128, a Commodore 64, amelyhez a könyv függeléke ismerteti az eltéréseket, és egy CP/M kompatibilis gép.

A kiadvány részletesen ismerteti a C128 BASIC V7.0 interpreter és a CP/M+V3.0 operációs rendszer használatát, majd kiegészítésként tárgyalja a BASIC 128 DATA BECKER fordító, illetve a JANE operációs rendszert.

Bemutatja a Commodore 128 két új funkcionális egységét: a 80 oszlopos monitornak és a memóriának kezelését végző MMU chipeket, azok specifikációit és felhasználásukat.

Az újszülöttek világa

A mesterséges intelligencia tudományos kutatásának és eredményeinek gyakorlati alkalmazása maholnap megteremti a hétköznapi feltételeket ahhoz, hogy valóságos döntési terekben emberi szempontok mérlegelését oldja meg a számítógép. Erre konkrét példaként most fejeződött be sikerrel Vámos Tibor akadémikus vezetésével az eddigi elméleti kutatási eredményeknek egy gyakorlati alkalmazásba való átültetése.

A kikísérletezett eljárás lényege, hogy az újszülött idegrendszeri sérüléseit igen korai szakaszban diagnosztizálhatják. Ennek során sok ezer adatot, megannyi formát, alakzatot digitalizált formában tárolnak, és logikai rendszerbe foglalva számítógéppel elemeznek, értékelnek.

A modell jelentőségét az adja, hogy az idegrendszeri sérüléseket már az újszülöttkorban megállapítják, sőt a gyógyítást is segíthetik; mindez sokszorosára növeli a gyógyítás hatékonyságát és a gyógyulás esélyeit. Az elért eredmények nemzetközi visszhangja igen nagy, és a nyugat-európai országokból is van már konkrét vásárlási szándékot rejtő érdeklődés.

Akupunktúrás portré

A moszkvai Fizikoterápiás Kutatóintézetben akupunktúrával gyógyítják többek között az asztmát és a neurotikus mozgásszervi megbetegedéseket is. Az orvosok úgy vélik, hogy ezzel távolról sem merül ki az akupunktúra alkalmazási lehetősége. További kutatásaikat segíti a modern technika, mindenekelőtt a számítógép. A legtöbb betegség ugyanis nem egy, hanem több pont ingerlése révén gyógyítható. A számítógép szerepe, hogy megrajzolja a beteg akupunktúrás portréját, vagyis hogy — az előzetes vizsgálati adatok betáplálása után, az orvosok és számítógépesek együttműködésével készült célrendszer lefutásával — meghatározza, mely pontokon és milyen intenzitással kell az illető személyt a tűkkel ingerelni.

Beruházások Dél-Koreában

1130 millió USA-dolláros beruházással fejlesztik 1986–87-ben Dél-Koreában a félvezető-technológiát és bővítik a termelést. Ennek eredményeképpen a félvezetőexport az 1985-ös egymilliárd dollárról 1988-ra 3 milliárd dollárra növekszik. A tőkét a dél-koreai Nemzeti Beruházási Alap adja. Az ország Közgazdasági és Technológiai Intézetének előrejelzése szerint a dél-koreai félvezetőexport a 2000-es években évenként 12 milliárd dollár is lehet.

A dél-koreai társaságok, amelyek a nyereségüket még főképpen a nagy bonyolultságú memóriák termelésének köszönhetik, berendezkedhetnek a világ félvezetőpiacának várható hosszú távú növekedésére, amely a 2000-es évekre kb. 300 milliárd dollár értékűre becsülhető.

Amit Barcs tanácsol: jó példa a tanácson

Professzionális személyi számítógép segíti az ügyfélszolgálati munkát Barcs Városi Tanácsán. Valamennyi, a tanácshoz beérkezett vagy onnan útjára indult levelet, ügyiratot begépelnek a mikrogépbe, sőt a tanácsülések, vb-ülések jegyzőkönyveit is. Ennek célja elsősorban az ügyiratok útjának nyomon követése, az ügyintézés határidőtartásának, eredményességének figyelése.

A számítógépes alapnyilvántartásokat, így a lakosság személyi és ingatlan-nyilvántartását jelenleg alakítják ki: elvégezték már a munka egyharmadát. Nagyon részletes, minden tárgyszerű információra kiterjedő nyilvántartást visznek fel az állandó lakosokról és a város területének valamennyi ingatlanáról. Egyelőre ezek az adatok jobbára a testületet és az apparátust szolgálják, de már keresik a jogi lehetőségeket arra, hogy a lakosság is meríthessen e hasznos forrásból. A következő lépcsőben a közintézmények és az adófizetők speciális adatai kerülnek a gépbe. Ennek elkészültével létrejön egy olyan információs rendszer, amely megkönnyíti majd a város közigazgatási munkáját, a tanács ügyintézését.

Hungarocard

Az év elején jelentették be — az SZKI az IPV-vel közösen — a Hungarocardra keresztelt hazai mágneskártyarendszer megjelenését. A rendszer a Proper—16 számítógépre épül, és a klasszikus gépen kívül egy leolvasókészülék, egy kódoló tartozik hozzá, valamint természetesen maguk a mágneskártyák. Ezekre a tulajdonos fényképe mellett mintegy 250 karakternyi információ fér el, ami például a tulajdonos nevének, személyi számának, címének felírására elegendő. A Hungarocardra épülő Cardio nevű vállalati személynyilvántartó rendszer segítségével nyomon követhető például az is, hogy a cég dolgozói éppen milyen üzemrészben tartózkodnak, a már hagyományosnak tekinthető belépés-nyilvántartáson kívül. A rendszer gépkocsik, targoncák és más munkaeszközök azonosítására is alkalmas. A Hungarocard ebben a formában hitelkártyaként nem felel meg, de áruházi, ún. törzsvásárlói nyilvántartás kialakítására alkalmas. Ez azonban csak akkor kifizetődő, ha több tízezer vásárlót lehet bevonni a rendszerbe.

SZAB(?) — szabászat

A Szombathelyi Háziipari Szövetkezetben számítógépes szabászati előkészítő rendszert vezettek be. Ez hatékonyan támogatja az új modellek tervezését: képes sablont rajzolni, méretezni, terítékrajzot készíteni, kiszámolja az anyagfelhasználást. Az előkészítés önállóbbá, gyorsabbá vált a felhasználóbarát géppel; a számítógéphez nem értő dolgozó is egy hét alatt megtanulhatja a kezelését. A tervek szerint a szövetkezet jövőre már külső megrendelőknek is készít modelleket a számítógéppel.

Generációváltás?!

Már harmadszor rendezték meg a Dunántúli Napló számítástechnikai versenyét négy megye középiskolásai és középiskolái számára. Az írásbeli első forduló alapján 11 résztvevőt hívtak meg Pécsre, az egész napos egyéni döntőre. A versenyen Gyuk Zsolt, a zalaegerszegi Csány László Közgazdasági Szakközépiskola negyedikes tanulója nyerte a fődíjként felajánlott ZX—Spectrum mikrogépet. Kiderült, hogy iskolája az első a Dunántúlon, ahol a gyerekek tantárgyi keretek között foglalkozhatnak a számítógépekkel; és ez meg is hozta az eredményt. Ez a győzelem is hosszú évek munkájának a gyümölcse, hiszen Zsolt már az első osztálytól kezdve „könyörtelenül műveli" a számítógépeket. Az „arany"-nak nagyon örültek tanítványai is — ugyanis ő vezeti a gellénházi általános iskolában a nyolcadikosok számítástechnikai szakkörét.

Az iskolák versenyében a zalaegerszegi Zrínyi Gimnázium csapata szerezte meg az első helyet.

Vásári kitekintő

Nem lehet elég korán kezdeni!

Nemcsak április, de május is szeszélyes, szélsőséges időjárásával maradt emlékezetes. Talán a tavaszi BNV szervezői, kiállítói izgultak leginkább, közönséget vonzó, avagy távol tartó napok köszöntenek-e rájuk? Hát végül volt ilyen, meg olyan is... A nem ideális körülmények ellenére eredményesnek könyvelhették el az 1987-es tavaszi BNV-t.

Szakmai vásáron valamennyi bemutató nem érdekelhet mindenkit. Speciális termékeknek speciális a közönsége. A megannyi látnivaló közül például sokakat csak a számítástechnika érdekelt, s akadt is bőven mit csodálniuk - de ez vonatkozott a többi profilra is.

A látogatót mindjárt induláskor érhette az első meglepetés, akkor, amikor el kellett döntenie, hogy a félszáz pavilon közül melyikben lel rá a keresett termékre. Tájékozódásában a számítógép segített: az eligazodást adó információként ugyanis printertől kapta meg a keresett helyet, a kiállítókat, a termékcsoportokat, mindezt címmel, névvel, telefon- és telexszámmal. (Jó, kétes!) Ott a technika, ahol a technikát bemutatják...) A kinyomtatott válaszokból tehát pillanatok alatt megtudható volt, hogy három pavilonban kell keresni a számítástechnikai termékek bemutatóját, hogy önálló épületben fogadja érdeklődőit a Novotrade Rt, s hogy további kiállítóhelyek is otthont adnak egy-egy számítógépes vállalatnak.

A Videoton, a SZÁMKI, a SZÁMALK és még jó néhány kutató, fejlesztő intézmény búvelte ki a K pavilont. Álmennyezetként és térelválasztóként vörös drapériát húztak az egyes kiállítók közé. Az enyhén rejtelmes félhomályban kitűnően érvényesült a jó helyi megvilágítás: a látványnak is volt köszönhető, hogy szinte ember ember hátán, hatalmas zsúfoltság. Az élelmesebbje mégis odaférkőzhetett a "tűzhöz", s kérdésekkel ostromolhatta a termékbemutatók szakembereit. Az ifjabb korosztálynak, a számítógép első-szerelmeseinek is akadt látni- és játszanivaló. A Videoton új termékét, a TV-Computer 64 k-t a helyszínen sokan "tesztelték", kipróbálva az ötletes játékprogramokat. A géppel a gyártó ugyan nem nevezett be a BNV oklevélre, mégis sikert arat majd. Annál is inkább, mert itthon ma már csak a székesfehérváriak gyártanak ilyen eszközöket. Méghozzá 11 ezer forint lesz a bolti ára (floppy nélkül), és így a szorosabb költségvetésű családokban is elérhetővé válhat a számítógép.

Míg a függönyök mögötti térben komoly üzleti tárgyalásokat, a nemzetközi tájékozódás külsőségeit is jelző neszek ígértek eljövendő sikereket, a fiatalok sorfalat alkotva várták, mikor kerülnek sorra, hogy ott és akkor kipróbálják a színes képi megjelenítésű számítógépet.

Az egyiknél hosszú hajú, serkenő bajszú ifjú nyomogatta fáradhatatlanul a billentyűket, feszülten igyekezve a szánkán sikló imposztorokat a kéz-, láb- és nyaktöréstől minél tovább megmenteni. Rövid beszélgetésünkkor is jobbára a képernyőre meredt, nagyon vonzotta az új "szerkezet". Bemutatkozásként elmondta, hogy Szántó Lajosnak hívják, a KIPSZER raktárosa.

- Nem ez az első találkozásom a számítógéppel. Imádok játszani a bátyám Spectrum 48-án, de azért a Videoton TV-Computer egészen más. Programozni ugyan nem tudok, játszani viszont igen. Ez egyszerűen megunhatatlan. Tetszenek nekem ezek a PC-k, küllemre is gusztusosak, kicsik és érthetőek. (Lám, mit tesz a formatervezés!)

- Te haver! Nem is tudsz szánkózni, azért rontod el olyan gyakran a játékot... - furakodik közénk egy nyakigláb kamasz.

- Hadd játszam most már én is ezen a gyönyörű gépen! - próbálja magának kiudvarolni tőlem a gép feletti parancsnokságot Megyeri József. (Ráhagyom...) - Egy új konstrukció kipróbálására a játék nagyszerű alkalom. Gimiben tanulok estin, napközben meg a Kelet-Pesti Vendéglátóipari Vállalatnál dolgozom mint kisegítő. Otthon - sajna! - egyelőre nincs még számítógépem, de már jól kezelem a Commodore 16-ot és a 64-et. Jártam egy BASIC tanfolyamra, sőt vizsgát is tettem abból a számítógépes nyelvből. Hogy miért tanulok ilyesmit? Na de hát ez természetes. Még fiatal vagyok, előttem az élet... És a jövő elképzelhetetlen számítógép nélkül! Meg azért, ha pusztán szórakozásra, játékra használja is az ember a technikát, de mégiscsak használja, együtt él vele, és nem az utcán csavarog... Ez kérem, olyan szórakozás, ami mellett észre sem venni, hogy repül az idő.

Harkányból Nagy Csaba, hetedikes tanuló egy másik asztalnál éppen megkaparintotta a gépet. - Most jöttem a Novotrade pavilonjából: lenyűgöző az a tengeretnyi számítógép, monitorokon követhető játékprogramvariáció! De ott csak a legerőszakosabbja próbálhatta ki a Cl6-ot és a C64-et. Mert például engem elzavartak az egyik számítógéptől - mondja kicsit rezignáltan. A legtöbbet csak a szemek nagyják, a kéz már nem élvezheti... Szerintem sokkal jobban szervezte, rendezte itt a dolgokat a Videoton. Most, hogy kipróbálhattam, én is szívesen hazavinnék egy ilyen gépet.

Az A pavilon szintén tömve érdeklődőkkel, s legalább annyira zsúfolt is látnivalókban. A szocialista kiállítók és a nyugati világhírű cégek kínálata mellett további magyar vállalatok, kis- és nagyszövetkezetek eredményeit tekinthették meg a kitartó érdeklődők. Több díjnyertes vásári termék között itt mutatták be a Kontakta gyár büszkeségét, a **Kodex 2000 szövegszerkesztőt** is.

A harmadik nagy számítástechnikai centrum a D pavilonban volt, ahol szintén sok nagy múltú, nagy nevű cég, vállalat jelentkezett legújabb, szemet gyönyörködtető s minden bizonnyal kelendő termékeivel. Kétségtelen, hogy sokan egyáltalán nem értették az adott szerkezetek működési elvét, nem ismerték az alkalmazási területeken várható hasznosságot, nem tudták elbírálni a fontosságát, de magát a tárgyakat, a mozgást, a színeket is élvezték. Így voltak ezzel a rádi nyolcadikosok is: Bodai Balázs, Csonka János és Deák Roland, akik osztálytársaiktól leszakadva - ne szépítsük: elcsámborogva - cikáztak a három pavilon között. Mint mondták, tájékozódnak, figyelik a technikát, a különlegességeket. Jegyeznek, érdeklődnek, kutatnak a márkák, a rangos számítógépek világában. Iskolájukban ugyan még nem otthonos a számítógép, de egy kihelyezett TIT tanfolyamon megtanulták a BASIC nyelv fortélyát, mondván: a középiskolában nem kívánnak értetlenül ülni az órákon. Különben is, ma már egy gondolkodó ember - vallják - a pénztárcájától függetlenül nem lehet meg alapfokú számítógépes tájékozottság, gépismeret nélkül. A vásáron ezért is tanulmányozták olyan nagy előszeretettel a jelen és a közeljövő technikai csodáit. Márpedig ezekből bőséges volt a kínálat...

Ugyanígy vélekedett egy sápadt, kisnövésű kamasz is, Domokos Tamás, kire a D pavilonbeli bolyongása közben bukkantam.

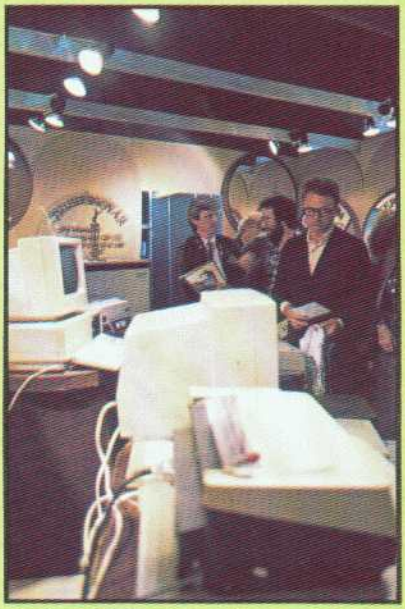
- Egyszerűen lenyűgöz a megszámlálhatatlan látnivaló. Amikor nincs szakmai nap, és a suli engedi, a vásáron vagyok. Nem tudom hányadszor járom körbe, a mindig felfedezek valami újabb érdekességet. Van otthon egy Videoton 32 k-s gépem, azon tanultam meg szeretni és becsülni a technikát. Asztmás vagyok, sokat hiányoztam az iskolából. Gyakran zárul rám a lakás ajtaja, és ilyenkor azelőtt nagyon unatkoztam. Egy beteg gyerek számára a számítógépnél idealisabb szórakozás el sem képzelhető... Aztán, ahogy túltettem magam a játékprogramok bűvöletén, könyvekből megtanultam színprogramok írását, vonalszerkesztő programok készítését. Most már tudom, hogy a számítógép elsősorban nem játék, hanem olyan munkaeszköz, amelyik társul is szolgálhat. Ügyes programokkal matekot, fizikát, kémiát és humán tárgyakat is könnyebben tanulhatok. Nos, testvér hiányában egy gépben leltem jóbarátra...

Nekik még játék és tanulás, a felnőtteknek sokszor mags a már szigorúbb élet.

KRASZNAI ÉVA

(Ez az oldal a BNV-díjas **Kodex 2000 szövegszerkesztővel** készült.)

A legújabb IBM számítógépek



A sztár, a Telefongyár nagydíjas terméke

A Kontakta BNV-díjas Kodex 2000-es szövegszerkesztője

Az NDK Robotron kinyomtatója

MINDEN,
AMIT TUDNI KELL
AZ IBM PC
XT/AT-RŐL!
számszöv
SZÁMÍTÁSTECHNIKAI KISSZÖVETKEZET
Cím: Bp. XI., Hunyadi János u. 162. 1116. Levélcím: Bp., Pf.: 16. 1430.
Telefon: 665-322